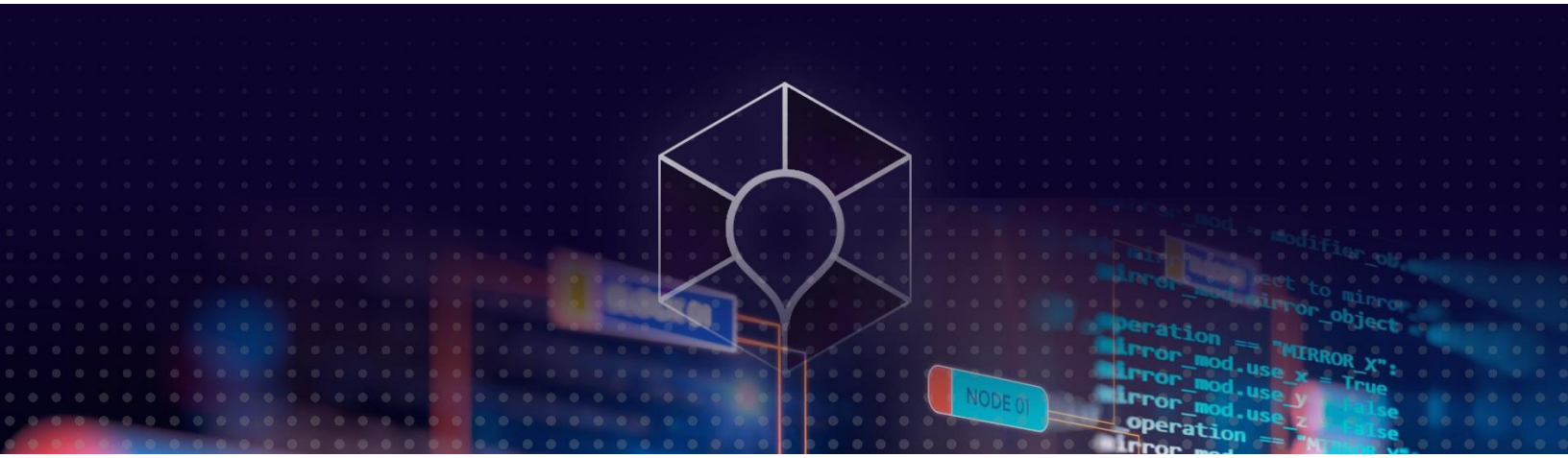
SIMULATING THE DIGITAL PRODUCTION PROCESS OF A HYBRID BIG DATA CONSULTANCY AGENCY IN KOTLIN

Master of Science Thesis
Industrial Engineering & Management

INFORMATION RESOURCES INCORPORATED
UNIVERSITY OF TWENTE



Émile G. S. G. Heijs

Manufacturing & Supply Chain Management

UNIVERSITY OF TWENTE.

SUPERVISORS

University of Twente
First: DR.IR. W.J.A. VAN HEESWIJK
Second: PROF.DR. M.E. IACOB

Information Resources Inc.
DRS. D. ROOST (Director of Technology & Operations)

ABSTRACT

The company Information Resources Incorporated (IRI), a big-data consultancy agency in the fast-moving consumer goods sector, faces serious efficiency problems in their operational department regarding the throughput time of data, workload deviation between employees and insight into the complex and abstract operational processes. The operational department of IRI is responsible for converting raw transaction data (containing a product code, units sold and a transaction amount which is obtained from retailers) to categorised databases that can be used to visualise and gain insight into this data. This research investigates the added value of a set of process modifications that could increase the efficiency of the operational department, which are defined based on an in-depth analysis of the key activities of the operational process. These key activities include: assigning a product category to the product code, assigning attributes to this product code, placing the classified product code in the right database and submitting the modified database to the IRI mainframe. This research implements a Discrete Event Simulation (DES) application in the Kotlin (a modern Java Virtual Machine) programming language. With the application of the Kotlin language, we aim to provide a future-proof and practical solution for management to gain quantitative insight into the production process. We use the DES application to evaluate the set of process modifications to both propose valuable improvements to the operational processes and offer a foundation for future research into several areas of interest substantiated by model results. The model makes use of a variety of historical internal process data. Results of the model show a number of modifications that could improve the efficiency of the operational processes:

- A workload assignment rule that assigns workload to employees by ordering the jobs based on the occurrence of product categories in historical data could decrease workload deviation between employees by 50% to 75%.
- Setting up a separate team that handles the assignment of new product codes to a product category could increase total jobs submitted by about 2% without affecting the performance of the rest of the system. This would mean a “free” increase of throughput.
- Increasing the “raw” data delivery moments from once a week to 2 times a week, smoothing out the arriving workload could decrease the throughput time by about 10% while decreasing the workload of employees by 10%.

Follow-up research is needed to increase the accuracy of certain aspects of the DES model and better investigate more complex aspects of the process such as new retailer integrations. With the implementation of the operational process of IRI in a DES model using the Kotlin programming language, we have been able to highlight promising modifications to improve the efficiency of the production processes. The research complements existing research in the analysis of stochastic production processes using DES by applying the technique to an abstract and digital manual production process in an unconventional way. Also, by using the Kotlin language and showing its functionality outside of the domain of Android development where it is predominantly used at the moment, we show a new side to DES research. Besides this new use of the Kotlin language, we also show how an existing open-source simulation library can be implemented effectively to a complex production process. This research can furthermore be used as a framework for the development of effective and practical DES applications for production system analysis using open-source DES libraries for general-purpose programming languages.

TABLE OF CONTENTS

1 - INTRODUCTION

This chapter will introduce the company in combination with a motivation for this research.

1.1 - The company

IRI (Information Resources Incorporated) Worldwide is an American data company operating in the fast moving consumer goods (FMCG) sector, with a strong presence in Europe. IRI Worldwide is one of the main providers for big data related services and consultancy worldwide and in the Netherlands. It is a mostly independent subsidiary focusing on the FMCG sector in the Netherlands, and is located in Zaltbommel. IRI obtains transaction data records (consisting of an European Article Number (EAN), a number of units sold and a total transaction amount) from retailers in the Netherlands such as supermarkets (Jumbo) or online vendors (Bol.com). This "raw" data is obtained either by helping the retailer process its raw data and obtain insights from it, or the data is bought. After receiving this data, it is processed each week in several steps by the operations department from raw records to structured databases. Clients of IRI (including retailers, but more importantly manufacturers of products such as Heineken or KraftHeinz) pay for this processed data in the form of custom databases, market research reports or insights. What a client receives specifically is agreed to in their specific contract. The core activities revolve around three areas:

- Unstructured raw data processing, where each week transaction data consisting of millions of EANs and a sales price are inputted into the IRI mainframe and processed by the system into their respective databases to be analysed. This happens automatically for those EANs already present in the system.
- Data enrichment, where EANs not present (known) in the system (but can be found on retailer websites) are processed manually (see Figure 1) and classified with a product category, a number of attributes (see Figure 2) and put in the correct database. This only has to be done once unless the product associated with the EAN changes. This is the responsibility of the operations division of the company. See Figure 1 for an overview.
- Data analysis. The transaction data combined with information obtained from the data enrichment step is visualised and analysed for paying customers in the form of reports (either in a file sent to the client, or via an online application called Unify) or to answer specific questions from clients. This is the responsibility of business analysts and technical consultants.

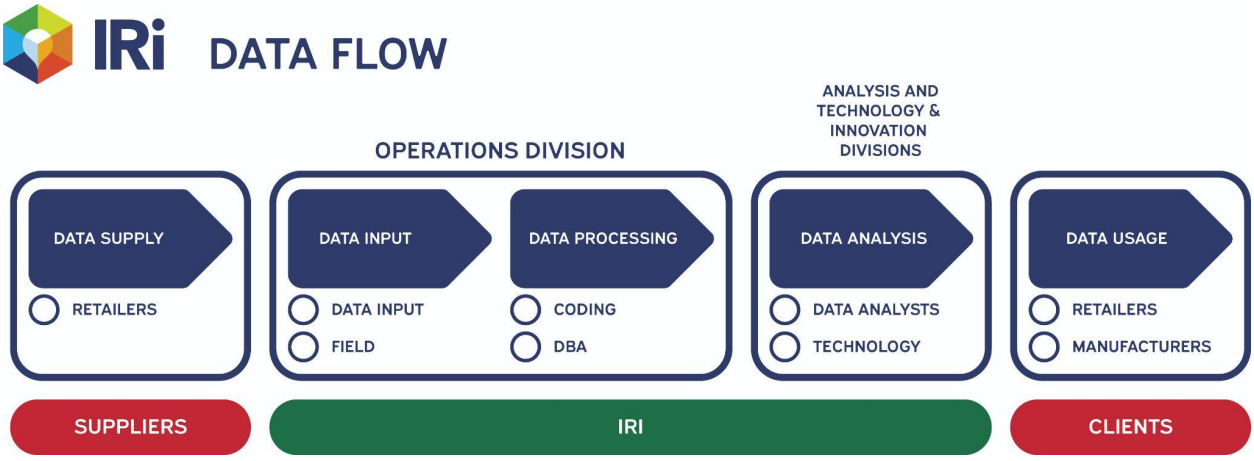


Figure 1 - Data flow diagram

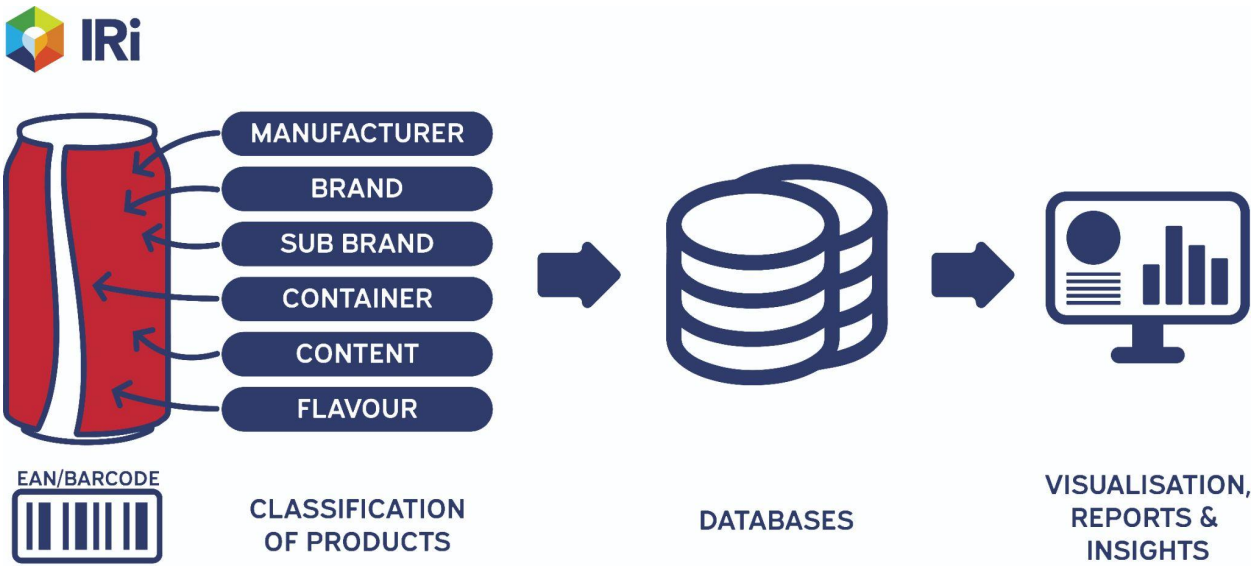


Figure 2 - From unknown EAN to client value

In the words of IRI, they are a data production consultancy company with raw data as input and structured data, knowledge and client reports as output. IRI is a modern, complex and dynamic company where improvement is embedded in the company culture. Constantly problems are identified and tackled. The company has a flat structure and a number of different departments as can be seen in Figure 3. However, being a data/tech company, operations research expertise is somewhat lacking. Management thought it therefore relevant to have the company, in particular the operations division, examined with a different view.

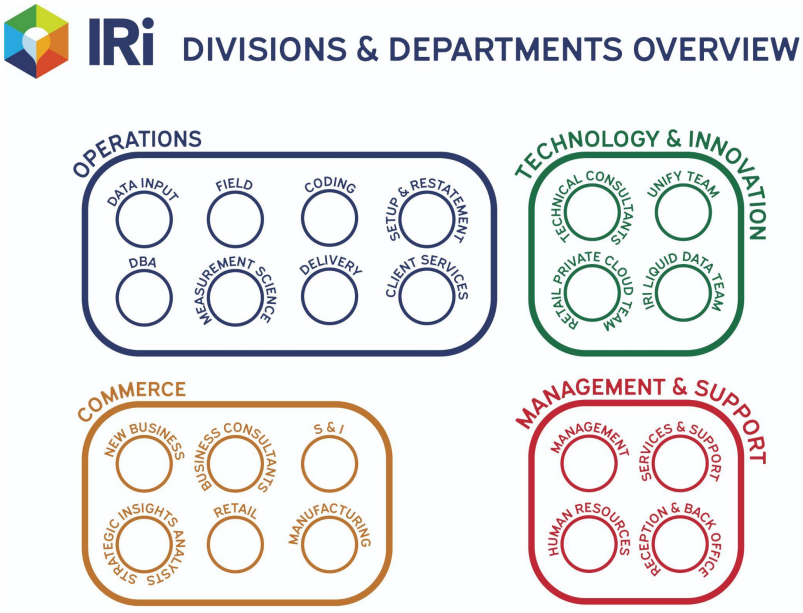


Figure 3 - Overview of company structure

1.2 - Research Motivation

This Section will give more in-depth information on the company and shortly explain the operational processes.
For IRI, having as much structured data as possible in the form of classified EANs with a category (Keycat), attributes and assigned to the right database is key. This is vital for their position towards their main competitor NielsenIQ, the correct functioning of the rest of the company and their value proposition for clients. A high coverage (> 98%) of a certain product category (Keycat) and the on time availability of new weekly transaction data in a respective database are the most important factors that decide whether paying clients join and remain at IRI. Companies producing products in the FMCG sector have an interest in gaining a timely and complete insight into the total market of one or more product categories. IRI has this data, and IRI provides them with tools and services to gain this insight in the form of category analysis reports, (online) visualisations, answering specific market related questions or giving advice based on the data available at IRI. The client pays for a certain package, containing a predefined number of (custom) databases, periodical reports and/or access to IRIs latest online visualisation platform and technology. What and how many of the services mentioned above are included depends on the contract.
The total collection of classified EANs in combination with (historical) transaction data is commonly referred to as the IRI Data Block (or the total database available on the mainframe). Looking to the future, this "data block" is estimated to be the key value proposition for at least the coming 5 years. With this in mind, one of IRIs key activities in the coming years is the integration of several new retailers that deliver their retail data to IRI. At the moment of writing, the integrated retailers delivering raw transaction data records to IRI include all (normal) supermarkets, drug stores, liquor stores and tank stations in the Netherlands (such as Jumbo, DA, Gall & Gall, Sligro) and some online vendors including Bol.com. New retailers integrations will include (smaller/lesser known) food retailers and more importantly non-food online retailers such as Amazon.
A transaction record received from a retailer can either be an EAN code that already exist in the IRI mainframe (it was processed before) and flow to the right (existing) database automatically, or an EAN code that does not yet exist in the mainframe and is therefore unknown (which means no category (Keycat) and attributes have been assigned to the EAN code). Normally these unknown EANs are new product introductions by retailers. In the case of a new retailer integration scenario, there exist two different possibilities:

- A product sold by the new retailer is the same as sold by another retailer which is already integrated, and as such has the same EAN code. If this EAN is already processed by IRI and therefore known in the mainframe, no further processing has to be done. If it is not yet processed it will depend on the total market turnover of this product if it will be processed.
- A product sold by the new retailer is not the same as any of the products sold by integrated retailers. In this case it will either be processed or not based on the market turnover for this product.

Such a new retailer integration can involve a very high number of unclassified EANs entering the IRI system, especially when the retailer operates in a different field than existing integrated retailers of IRI, this has for example been the case with Bol.com where the discrepancy between food and non-food products is quite high. A percentage of incoming unknown EANs (determined on sales volume) will enter the operational department of IRI to be processed and become the workload of the department.
As IRI has had few new retailer integrations for a long time, the process is designed to perform under relatively well known normal workload conditions processing the new product introductions of already integrated retailers. The recent integration of Bol.com has learned that the necessary move towards more regularly integrating new retailers can strongly disrupt the operational process. The drastic spikes in arriving unknown EANs and the workload associated with such new integrations have long been unknown in the company.

Solutions as increasing capacity are not always desirable, as these workload spikes are temporary and hiring (and training) new employees not only costs resources but also decreases efficiency of the existing workforce. On top of this the work has a high level of complexity. This can be attributed to a high number of government regulations and agreements with clients that need to be followed, exceptions to these rules or related to the market condition/period of the year and the complexity of software systems used. Because of this, training new people takes a lot of time. Adding to this, high variation of the workload can lead to highly unequal workload division which again increases work strain (also on existing employees) and sometimes effectuates people getting overworked and/or quitting.
Some ideas already exist that could improve the efficiency of operations. However, little is quantified concerning both implemented and not yet implemented improvements of the process. Furthermore, a concrete, complete and detailed overview of the operational processes does not exist. This makes it difficult to substantiate decisions and implement meaningful improvements. Therefore, for this research, the operational processes need to be mapped and analysed in detail, and quantitative insights in process and possible improvement needs to be enabled.
In the situation outlined above, two problematic aspects can be identified namely:

- The lack of insight into the operational process (complete detailed process overview, quantifiable effects of process changes, performance of the process under changing conditions)
- The degree in which different factors of variation that occur in the process negatively impact efficiency (varying job input amount, varying workload per job, varying number of employees)

In the domain of Operations Research there are a multitude of techniques that can be used to model a process in a stochastic (we will consider factors of variation in the model, see Section 5.2) problem context where we are interested in getting a better understanding of a stochastic process and the performance of process alternatives. One, in the literature generally considered the most practical and effective, is Discrete Event Simulation. It is currently used in many professional environments for its ability to represent processes accurately and provide a high level of credibility because of the relative closeness of the model to reality. We also expect that this approach, when using a modern general-purpose programming language such as Kotlin (a modern alternative to Java) and open-source simulation libraries could provide a high level of model resemblance to the real process and provide opportunities to make the model interactive and usable outside of this research. A deeper analysis of these techniques including a final selection of technique and programming language can be found in the literature review of chapter 4.

2 - PROBLEM ANALYSIS

In this chapter, we will analyse the company further to find out what can be improved. We will examine problems and opportunities that are present in the company and distil a suitable core problem.

2.1 - Problem Context & Causal Problem Diagram

At the start of this research IRI had not provided a concrete assignment or problem context, therefore a better understanding of what happens in the company is in order to finally determine the most relevant core problem. To get a good overview of the complete problem context and everything else going on in the company, we have held interviews with employees from a multitude of different departments. To get a view of and experience in the operational activities of IRI, we have held in-depth days viewing (and partly performing) all activities of every operations department. This helped with getting an objective view of each department (not influenced by the existing views within the company or opinions of other departments/people) We constructed a list of every identified problem or opportunity, separated per department, which can be found in appendix A. Consequently, we analysed cause and effect by making a causal problem diagram presented in Figure 4 on the next page.

2.2 - Core Problem Selection

This section will give a motivation for the selection of the core problem to be solved following from the problem context defined in chapter 3. The core problem will be selected by defining the most important processes within IRI, elaborating on the specific challenges that have been identified within these processes. Based on these first steps the most relevant and important core problem can be selected.

2.2.1 - Key processes

There are multiple core problems that can be chosen from the causal problem diagram. To make sure the core problem chosen is the most relevant one we have identified three key process flows within IRI. These are either essential to the parts of the company contributing to client value creation or do this directly (such as providing visualisations of a category market or providing analysis reports to clients):

- The data processing of new EAN codes of integrated retailers (data suppliers) - defined here as the base workload
- The data processing for new retailer additions - defined here as the overflow workload
- Gaining meaningful insight from processed data (report production, (online) data visualisation, application of machine learning)

The reason for choosing these three processes is because of the following: The processes within IRI that directly add client value such as report creation, data analysis and visualisation all depend on the smooth flow of the operational processes that are responsible for the processing of new (and unknown) EANs. Although the first two points have been defined as two separate processes, they both relate to the processing of new unknown EANs. This processing is done by two departments within the Operations division, namely Coding and DBA (Database Analysis). Coding is responsible for the assignment of the right product category to an EAN and then the assignment of a number of attributes (related to the product category) to the EAN. The EANs selected to be processed by Coding are determined by the expected addition to the coverage of turnover of a certain product category. DBA is responsible for the correct Placement of the newly coded EANs into each client database. Steps include verifying these “new adds” are placed in the correct category database (which happens automatically) and if attributes are at the right database level.

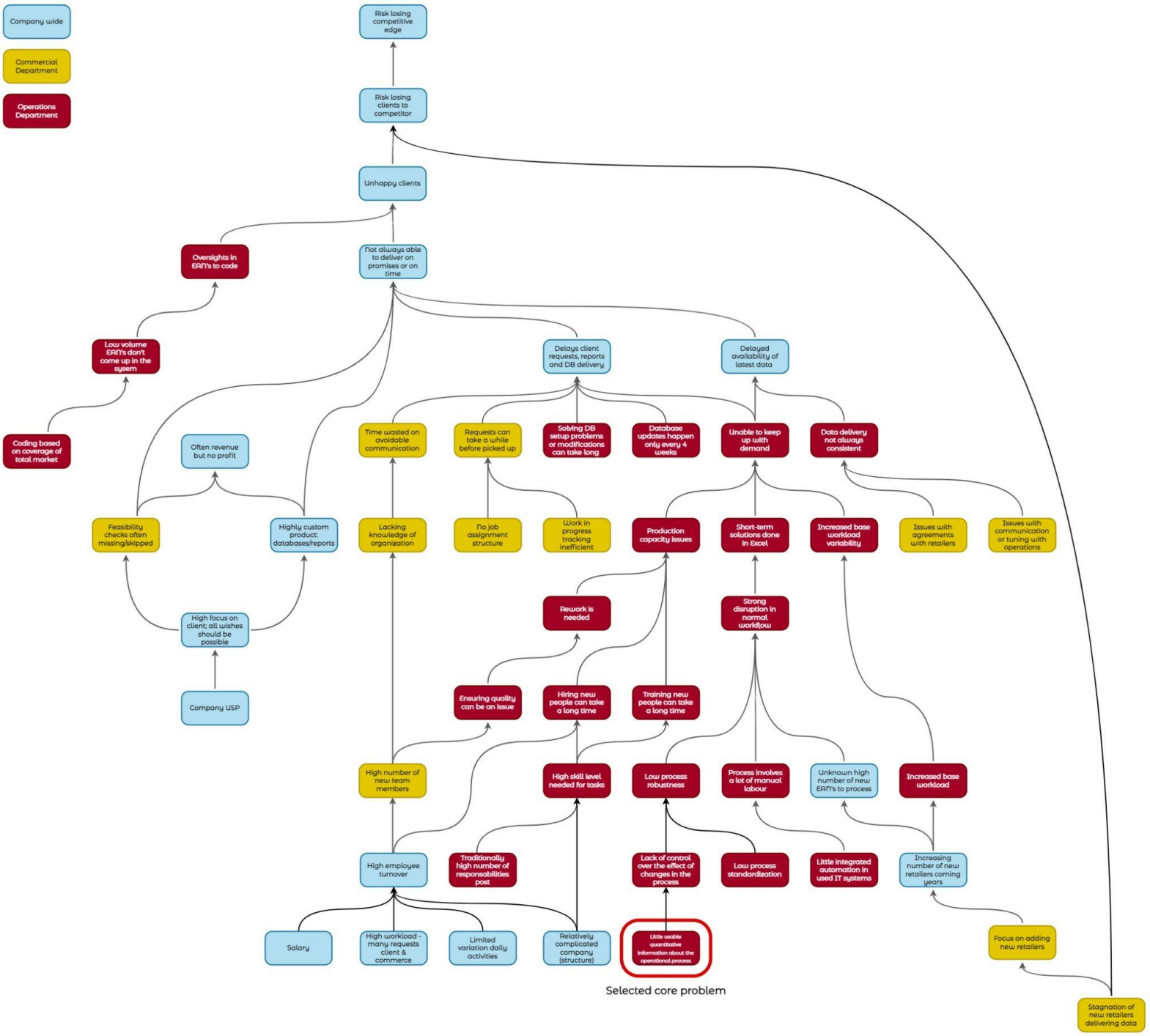


Figure 4 - Causal problem cluster

Factors of success of the processes related to direct client value creation are:

- On time data delivery
- The avoidance of delays (in data availability, database delivery and report delivery)
- Keeping a competitive edge
- Satisfied employees
- Satisfied clients
- Gaining new clients

The current vision of IRI (high focus on customizability and client wishes) and the expectation of a lasting trend of new retailer additions (also possibly including a move into Belgium), leads to the conclusion that in the current climate of the company the operations division needs to adapt to be able to operate more efficiently, effectively and in a more structured way.

2.2.2 - Operational Challenges

A number of operational challenges arise within the defined core processes defined in subsection 2.2.1, which will be elaborated now. The base workload of the Coding and DBA departments depends on the number of new product introductions entering the assortment of an existing retailer each week. A number of these new EAN codes entering the process of IRI have a high enough importance (consisting of a number of factors such as sales volume and if the producer is a client of IRI) to be processed by the operations department. Processing means a number of things:

- Assignment of a Keycat (either by the Coding department or the Auto Engine Artificial Intelligence)
- Assigning a number of attributes (corresponding to the Keycat) to the EAN (done by Coding)
- Loading the new processed EANs in existing databases (done automatically based on certain rules)
- Verifying the new processed EANs have arrived in the right database, and the attributes correspond to the levels of the database (done by the Placement division of the DBA department)
- Sending the list of new additions to the database to a paying client for feedback (done by Placement Support)
- Processing the received feedback (done by Placement Support)
- Submission of the database so it can be used by the rest of the company for analysis (done by Placement Support)

This process is subject to high workload variation largely outside of the sphere of influence of IRI. Factors that contribute to this variation are:

- The number of different Keycats a unknown EAN can belong to (~580)
- Factors if and when retailers introduce new products
- The varying amount of attributes that need to be coded per Keycat
- The varying amount of databases new EANs need to be included in
- The amount of levels in a database that need to be checked by Placement (only relevant for ILD*)

* ILD (IRI Liquid Data) is a new database technology currently being implemented in the company. It is expected to not significantly impact the operational process.

In production systems, variance of base workload can (in time) lead to problems with production planning, workload distribution and it can have an impact on achieving the previously mentioned factors of success. In some parts of the company this workload variation already leads to problems, which leads to overworked, unhappy or quitting employees which leads to an only further expanding problem. In this case the variation of EANs coming in each period (product introduction at the retailer and the workload within IRI) is outside IRI's control, as IRI needs to process everything since this is expected by paying clients (data buyers or manufacturers).
IRI will also increasingly have to deal with the question of how to respond to the overflow workload involving the (partially) unknown amount and time of arrival of new EANs at a retailer integration. At the moment new retailer addition workload is not handled on a separate process, negatively impacting the capabilities of handling the base workload, which in turn negatively impacts the whole of the company. The fact that training new employees takes a long time and it is not known what direct effect these new employees have on the productivity forces current teams to apply temporary and inefficient solutions (involving a variety of Microsoft Excel tools and sheets) to cope with sudden extreme increases in workload (examples can be seen in chapter 6 on data analysis).
In conclusion, there exists too little quantifiable information about the performance of the current operational process in a changing environment and the direct effect of workload variation and modifications of current process parameters and possible improvements of the process. Quantifiable insight needs to be gained into the current process and process alternatives that could reduce the negative effects of workload uncertainty, increase efficiency, while also providing IRI information on the effect of hiring/firing employees, developing complementary solutions to improve the process such as web scrapers or workload forecasting models .

2.2.3 - Core Problem Definition

Following from the operational challenges that arise in the operational process within IRI as defined in subsection 2.2.2, the core problem selected for this research is:

There exists little quantifiable information about (the effectiveness of) the current operational process especially in a changing environment (of more common new retailer integrations) and the effect solutions can have on process performance for management to make substantiated strategic decisions.

We have chosen this core problem since we expect the largest gain can be achieved by solving this issue compared to other identified core problems. This core problem is linked with two core processes within IRI that allow for client value creation, as has become evident from subsection 2.2.1. With solving this core problem an increase in process robustness and quantitative insight into effects of increasing workload variance are expected to be gained, but also more general strategic insight into the modification of process parameters (such as number of employees, productivity levels or days on which activities are performed). Process robustness is defined as the manner the process flow is formalised and structured and the manner in which the process is able to handle and adapt to a certain degree of variation, uncertainty and the (unforeseen) change of variables. Robustness of a production schedule can be defined as the following: a schedule that performs well under real world scenarios/conditions, like capability to handle small delays, to resist imprecision, to tolerate a certain degree of uncertainty or to cope with unexpected troubles without significant modifications (G.E. Vieira - 2017, E.V. Andersson - 2014, M. A. Salido et al. - 2008, Policella - 2005, Takeuchi and Tomii - 2005). Quantitative insight is defined as a method that IRI can use to substantiate process oriented decisions with the help of a quantitative model of the process to determine the effectiveness of said decision.
We have identified other problems/opportunities that can be seen in appendix A and in the problem cluster of Section 2.1. Some of these are already being addressed such as the stagnation of new retailer integrations (which in their turn create new problems as explained in section 4.2). Others are in some way intertwined with the selected core problem

above, such as the high employee turnover in the Operations division (this is also an issue in some other divisions such as Client Services but are left out of the scope of this research). There are also "problems" defined that can be seen both as a problem and as an opportunity or positive factor for the company such as the company USP or things that cannot or are unlikely to change such as salary.

In conclusion, based on the steps taken in this chapter, a core problem has been defined that is expected to provide the most added value in the current state of research on the operational process within IRI. In solving the core problem, it will act as a basis both for management to make more informed decisions and provide a basis for further research into this department.

3 - THE RESEARCH SETUP

This chapter will define the research goal together with the expected output and give a scope.

3.1 - The Main Research Question

The goal of this research is to obtain an answer to the main research question:

Can a Discrete Event Simulation implemented in general-purpose programming language provide relevant quantitative information on process performance in the changing operational environment and provide management with the necessary insight to make better substantiated strategic decisions?

This research question is two sided. On the one hand, IRI needs information about the performance of the current process. In the changing environment IRI finds itself, more regular new retailer introductions are expected to increase the effects of variation and uncertainty in the operational process of the Coding and DBA departments (who are most strongly affected by the workload variation and overflow uncertainty). On the other hand, IRI needs to know if process alternatives could improve the performance and robustness compared to the current process and what these scenarios will entail. The result of this research will therefore be:

- An analysis of the current operational process and its baseline performance based on measures outline below
- An analysis of the performance of process alternatives (scenarios, consisting of modified process parameters or other process setups) that could improve the efficiency and flexibility of the operational process and provide management with the insights to make more informed decisions
- And a performance analysis in a scenario with a new retailer introduction
- A reusable modular process performance computer program

Each defined process alternative will be analysed on a number of key performance indicators (KPIs). These KPIs to evaluate the performance of process alternatives can be summarised in three main aspects of success namely: data availability, throughput capacity and workload deviation. The following KPIs are used for this analysis and there relation to the aspects of success are indicated:

- Job cycle time (time needed from barcode arrival to submission of the database) - data availability
- Number of unfinished jobs in the system - throughput
- Number of submitted jobs - throughput
- Number and standard deviation of jobs in the queue of a certain station - workload deviation
- Variation of workload per week of a production period (total system and per activity) - workload deviation
- Variation of employee workload (deviation) and utility - workload deviation

Based on percentage comparison between process alternatives with the baseline process performance, insight can be gained to serve as input for strategic management decisions concerning changes in the operational process.

3.2 - Scope

This research will focus on providing quantitative insight into the performance of a (data) production process and its alternatives in a changing environment which involves various kinds of uncertainty and variation outside the sphere of influence of the company. The scope of the research will be limited to the Coding and DBA departments (as opposed to

all operations departments) since they are most impacted by an increasing number and variation of barcode arrivals, they also are commonly regarded as the bottleneck of the complete operational process and consist of the highest number of employees. As mentioned in the research goal, the output of this research will be an analysis of the current process, an analysis of several scenarios that could help improve the performance of the operational department, also in a changing environment where new retailer introductions will be more and more common. This will most likely be achieved with the implementation of a Discrete Event Simulation (DES) model written in Kotlin (see Section 4.3 for elaboration on chosen technique). The model will focus on and include the number of unknown barcodes entering the system, job processing times, number of employees, rework (mistakes made and fixing them). The model will also exclusively focus on the activities evolving around the normal workflow of the two aforementioned departments on processing unknown EAN's and exclude any side activities or separate responsibilities. The final DES application will not include an intuitive graphical user interface (GUI), and only serve as a base model to analyse scenario alternatives. A more "barebones" user interface will be included to provide the opportunity to use the model via an executable file. The model will however be set up with modularity in mind, as well as the inclusion of an interactive GUI and additional extensions of the base DES model.

3.4 - The (Sub) Research Questions

To be able to answer the main research question, the research is divided into 6 phases each containing a number of sub-questions that will be answered in that phase.

1. Business Process Model of the existing process
    a. What does the current operational process within IRI look like, specifically regarding the main data processing flow?
2. Literature review & selection of research method
    a. Which state of the art Stochastic process modelling techniques exist in the Operations Research and which technique is most suitable for this research according to the literature?
    b. Which techniques can be used to solve the model obtained using the technique chosen in 2a to obtain the required research output defined in Section 3.1?
    c. Which program/library/programming language is best to create and solve the model and meet the requirements defined for the results of this research?
3. Data collection and analysis
    a. Which data is needed as input for the model?
    b. Which data is already available?
    c. Which data is not available and which solutions can be found for this missing data?
    d. Which period from the data can be used as a stable baseline scenario?
    e. How does a retailer integration differ from the baseline workload scenario and how can this be best analysed in the DES model
4. Model development
    a. How can the BPM of question 1 be used as input to model the process?
    b. Which elements of the techniques from 2a and 2b need to be defined to allow for correct development?
    c. What are the most important decisions made and characteristics of the implementation of the model?
5. Definition, integration and analysis of process alternatives
    a. How does the baseline process perform according to the model and which parameters are used?
    b. Which process alternatives will be calculated by the model that either could improve process performance or provide valuable process insight?

    c. What is the performance of these alternatives compared to the baseline performance?
6. Conclusions & recommendations
    a. What are the limitations of the model?
    b. What are recommended actions for IRI to take based on the results of the research?
    c. What conclusions can be drawn from the results and recommendations of the research?

In this chapter we used the content of chapter 2 as a basis to establish a definition for the setup of the research in the next chapters. Based on the main research question, derived from the core problem, we have defined the result of the research bounded by a scope. Lastly, we have drawn up the sub research questions that together will provide an answer to the main research question and also serve as a general outline of the steps taken in this research.

4 - LITERATURE REVIEW

To solve the selected core problem and achieve the research goal, the academic domain in which the problem resides will be explored in this chapter. The literature review will explore the academic field of several techniques that can be used to model and analyse a system that involves uncertainty, also known as stochasticity. In solving the core problem we are looking at techniques that can serve in increasing process robustness, providing insight into process alternatives to make better strategic management decisions and that can incorporate elements of process uncertainty.

4.1 Stochastic modelling methods for production systems

Based on the information gained from chapter 2 and 3 we have defined Discrete Event Simulation (DES) as a preliminary method to apply in this research because of the characteristics of the system and core problem. In this section, stochastic modelling and analysis methods will be explored more in depth to make sure the selected method corresponds well to the requirements and environment of the research problem. The technique which is finally selected should be able to cope with complex process modelling (including uncertainty) and evaluation while remaining easy to understand (for credibility purposes), easy to modify (for longevity purposes) and the computer application built using the technique has to be fast to execute (for practicality purposes).

4.1.1 - Types of uncertainty and modelling techniques in production systems

S.C. Graves (2011) identifies three types of uncertainty that commonly arise in a production systems, namely:

- Uncertainty in the demand forecast
- Uncertainty in the external supply process
- Uncertainty in the internal supply process

In this research the first type is irrelevant since IRI does not make workload or demand forecasts in the operations department. Uncertainty in the external supply process within IRI can be seen as the variation in jobs that arrive each week plus the uncertainty in overflow workload and uncertainty in the internal supply process as the variation in internal processing times.

There is no generally accepted list of stochastic modelling types applied in production environments. J. Mula et al. (2006), provide the following four types of methods based on their systematic literature review of used methods in production systems under uncertainty:

Table 1 - Classification for the general types of uncertainty models in manufacturing systems (source: J. Mula et al. (2006)

Conceptual Models	**Analytical Models**
Economic Order Quantity, Yield Factors, Safety Stocks, Safety Lead Times Requirements Planning	Mathematical Programming (such as Linear, Integer, Dynamic), Stochastic Programming, Deterministic Approximations, Markov Decision Processes
Artificial Intelligence Models	**Simulation Models**
Reinforcement learning, Fuzzy set theory, Fuzzy logic, Neural networks, Genetic algorithms, Multi-agent systems	Probability distributions, Heuristic methods, Freezing parameters, Network modelling, Queuing theory, Dynamic systems

P. Martinez & R. Ahmad (2021), differentiate between Analytical and Artificial Intelligence Models for manufacturing quality problems and extend this classification with simulation models in an inspection process planning problem environment. They disregard the conceptual models from Table 4, as they are generally not well suited to model more complex production systems. We will continue using this classification and explore some of the more common modelling techniques applied in stochastic production systems and determine whether they have added value to solve the core problem compared to Discrete Event Simulation.

4.1.2 - Simulation models

Simulation models can take a variety of different forms. A. Tako & S. Robinson (2012) show that simulation is often used in the domains of Production Planning & Scheduling, System Performance Analysis and Cost Reduction. In their literature review that analysed 256 articles in the period between 1996 and 2006, simulation is used in 25% of the articles over 17 fields. Regarding simulation, they conclude that from a variety of techniques, simulation modelling (Discrete Event Simulation (DES) and System Dynamics) is mostly used to model issues for in depth analysis and policy formulation. DES/stochastic models are predominantly used to study the detailed operation of an environment with the inclusion of uncertainty and/or to evaluate the expected performance measures to a high level of accuracy, W.L. Winston (2004) describes simulation as a technique that is highly suitable to model and analyse complex systems without the need to make many simplifying assumptions. Other advantages include flexibility in representing the real system, credibility and the ability to use a wider range of probability distributions. Winston also points out that, since simulation is not an optimization technique in its core, main applications are analysis of "what if " scenarios and the analysis and/or the comparison of scenarios. In existing research, DES is most prominently used in a manufacturing context, with a recent shift to more service oriented systems such as healthcare. This requires a stronger focus on the representation of human behaviour in the model. The inclusion of human behaviour is also relevant in a manufacturing context in an effort to provide a more realistic representation of a production system. Kampa, A., Gołda, G., & Paprocka, I. (2017) propose a model for a system where there is a choice between human operators and robots. The combination of a services oriented company combined with a production process, such as the process as IRI, has however not yet been explored in the literature. There is also room for improvement in model development, model use and the integration of the technique with other technology as Robinson (2005) points out.

A modelling technique that can be seen as a bridge between simulation and analytical models (explored next in subsection 4.1.3) is Queueing theory. A queueing model works, like a simulation model, with input/service distributions and servers (with a queue) in a network form. The analysis of the queueing network is done with the use of fundamental queueing laws in a mathematical fashion. Therefore, queueing models often require many simplifying assumptions and have a limited range of probability distributions that can be used as input (Winston - 2004). This makes the technique less practical, especially in more complex systems and less applicable to solve the core problem of this research.

4.1.3 - Analytical models

Analytical models employ fundamental mathematical laws to describe systems and are characterised by state- and solution spaces. Stochastic Programming (SP), an analytical method first introduced by Dantzig (1955), is the classic example of an optimization technique for stochastic Operations Research problems. SP is an evolution of Linear/Integer programming, where optimization functions are formulated for every possible scenario a system can be in with a certain percentage (making the system stochastic). When problems get larger the optimization (or enumeration/calculation) part is often very difficult since often (even relatively simple) system/problem formulations are not possible to solve in polynomial time on current computers (NP-Hard), especially when integer decision variables are included (M. Dyer, L Stougie - 2005). This means the solution space is so large, there does not exist an algorithm that can evaluate every possible solution to the system in a reasonable time-span using current computer architecture (which is described as

Non-Polynomial or NP). To be able to provide a solution to the SP formulation that approaches the optimal solution the use of alternative methods to explicit enumeration are required. An example of an alternative technique is the use of (meta)heuristics. As this technique makes use of algorithm logic to evaluate a subset of the solution space in a smart way, there is no guarantee an optimal solution is eventually found. Another disadvantage of SP is that the "scenario" formulation and output of the model can be difficult to communicate or implement. One of the results in practice of these disadvantages of SP is that the element of uncertainty is often omitted from the model and buffers such as safety factors, flexible capacity, inflated lead times or replanning are calculated to reduce negative effects the uncertainty has on the performance of the system (S.C. Graves - 2011).
Other analytical models often used to model manufacturing systems such as Dynamic Programming (R.E. Bellman - 1957) have recently evolved to be quite effective in handling classic problems in the field of Operations Research. S. Xu et al. (2020) show that more advanced (hybrid) applications of this technique, in the form of an Approximate Dynamic Programming model combined with Deep Neural Networks, can achieve significant computing time reduction on combinatorial problems. This tackles however only one inherent disadvantage of analytical models, and offers a difficult theoretical approach to theoretical problems.
One of the requirements of this research is to analyse the effects of uncertainty, we can therefore not omit this to simplify the model. In addition, we would like to model the process at IRI in such a manner that it is precise enough to gain more in-depth knowledge about the workings of the process and simple enough to remain practical in its running time for the company. An modelling aspect we can omit to arrive at a solution to the core problem is optimisation, something that is not possible with SP and DP but is with DES. This means that SP and DP do not provide us with additional value to solve the core problem in this research.

4.1.4 - Artificial Intelligence models

Artificial Intelligence based methods have become very popular in the last 20 years with increasing computing power and data availability. Also practical use in production environments is high because of their lower design complexity as these methods require less prior knowledge of the process in question and require mainly high quality data. The availability of data, that in recent years has surged (in production environments because of industry 4.0 for example), makes a strong case for the application of data driven approaches. In depth data analysis and data science approaches have gained significant attention from current research that can analyse complex relationships in big data sets. The fact however that this approach does not require a good understanding of the production system, does make it less applicable in this research as one of the requirements is gaining a better understanding of the system.

4.2 - Discrete Event Simulation

Section 4.1 has shown that DES is the most suitable approach to solve the core problem of this research. Other techniques that can be used to model stochastic production systems do not provide inherent benefits compared to DES. This section will provide background information, insight into evolution and the latest advances of the DES technique. The background information will serve as a starting point, provide definitions and inform readers not familiar with the technique. Insight into the evolution of DES will provide a better understanding of the research domain and serve as introduction to the section containing some of the latest developments in the domain of DES. These are included as they might be relevant to this or follow-up research.

4.2.1 - Simulation: concepts, characteristics and definitions

One of the first considerations when modelling reality in operations research related situations, is whether or not to include random (or uncertain) variables in the model. This provides the distinction between deterministic and stochastic models. As said, in general it can be stated that stochastic (analytical) models are computationally significantly more

complex, often to the point of being impractical in real applications. With the ever increasing power of computers, (stochastic) simulation has provided an effective alternative to stochastic analytical models. The second consideration relevant to simulation models is if it is static or dynamic. Static means the system does not evolve over time, dynamic means the opposite. The third characteristic of a simulation model is the distinction between continuous and discrete systems. In continuous simulation models the state variables can change and are monitored continuously while in discrete simulation models the state variables only change in set time intervals.

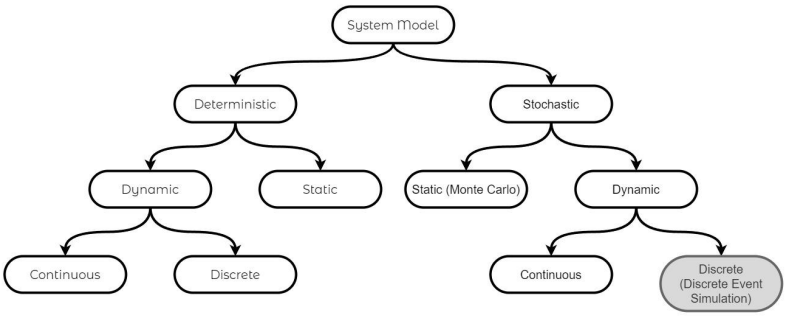


Figure 5 - Types of Simulation models

In operations research applications the majority of simulation models are stochastic, dynamic and discrete, also known as Discrete Event Simulation models. In such a scenario interest lies in a finite set of events that change the state of the system. An advantage of discrete simulation models are the significant computational resource savings. In a discrete event scenario, the type of time advance mechanism becomes of importance. Winston (2004) identifies two types of mechanisms: the next-event time advance mechanism (where the discrete time steps are determined by the time when the next event occurs in the event-list) which is computationally more efficient (minimum number of time jumps) and the fixed-increment time-advance method (where the discrete time steps are determined by a fixed time increment, often the shortest time period two events are separated from each other) which is easier to understand and implement. The next split in DES models lies in the simulation horizon. A simulation run can either be terminating or non-terminating, where in a non-terminating simulation run interest lies in the steady-state behaviour of the system. In a non-terminating simulation, to get a representative steady-state reading for statistical analysis the effect of the initialization bias has to be dealt with. Generally a simulation run starts empty (no jobs in the queue) which is not representative of the normal (steady) state of the system and therefore impacts the measurements. Regarding the measurements and statistical analysis of simulation results, the fact that measurement parameters can generally not be considered to be independent impacts the statistical methods that can be used in analysis of the output of the simulation system. Lastly, before constructing a DES model a primary objective has to be determined and the kind of probability distribution the stochastic variables in the model are sampled from. This sampling is in turn done by selecting a random number generation technique compatible with the probability distribution function.

4.2.2 - Evolution of DES

It is generally considered that the biggest and most significant advances in DES have taken place from its inception in the 1960s to its maturation in the 1980s with the availability of easy to use commercial simulation packages. E. Babulak & M. Wang (2010), for example, in their review of the current status of DES development. They note the absence of significant development in DES technologies after the 1980s, which they relate to the unprecedented changes in the manufacturing industry (one of the principal adaptors of simulation technology) as a result of globalisation. Nevertheless, DES has evolved to be one of the most effective and practical Operations Research techniques applied to a great range of (business) scenarios. Robinson (2005) makes some, remarkably accurate, predictions looking forward to

the future and development of DES which could potentially renew innovation in the DES domain. He expects a consistent further increase in computing power and developments in visuals and interaction with the model. In fact, he believes that both the increase in computing power and the continuing development of commercial DES software allows for significantly higher levels of visuals and interaction. In addition, where in the 2000s developers are used to writing their own code and develop in a structured manner, a new generation of developers will mainly reuse someone else's code which is found online, which they expect to be free, and search for code and development will happen in a more unstructured manner. Of course, with platforms such as Github and the rise of open source software, this prediction seems quite spot on.

An often mentioned evolution of DES is the use of 3D models and Virtual Reality visualisation and interaction. While this idea stems from the 90s, C.J. Turner et al. (2016) name the continuing growth in computing power and the more recent increase in computer graphics as important factors allowing the possibility of combining DES and Virtual Reality. The combination of the two techniques has as most prominent advantages that it offers decision makers a greater level of understanding and modellers a more thorough identification of modelling errors and omissions. One of the most apparent disadvantages is the longer development time. A related concept in recent literature is the term "Digital Twin" which is used more and more to describe the implemented (interactive) DES model (software). Digital Twin, a term associated with Data Science, Machine Learning and the Internet of Things, is a virtual representation of an object or process. Indeed, the demand for more complete representations of production processes has risen with cutting edge manufacturing innovations such as'Industry 4.0. An early example can be given by B. Kádár et al. (2004) who propose a Digital Factory solution for a planning and scheduling problem. R.G. Alves, R.F. Maia & F. Lima (2021) use the DES technique in combination with Internet of Things to provide irrigation prescriptions for farmers in Latin America. Although the evident advantages for Industry 4.0 companies can be seen, this route will be less relevant to the mainly digital production environment of IRI. Figure 6 shows the evolution of the manufacturing industry, the combination of simulation and virtual reality can be seen as a Cyber Physical System.

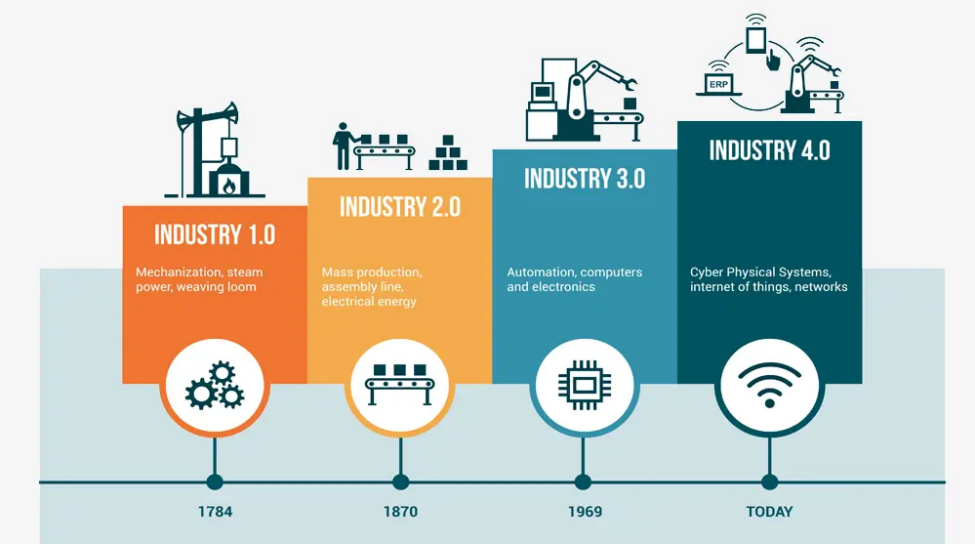


Figure 6 - Evolution of the manufacturing industry
(source: seekmomentum.com)

Above examples apply to traditional manufacturing environments, which is the case for many DES applications (S. Tavakoli et al. - 2008). X. Zhang (2018) shows an important trend with his systematic literature review covering 220 articles in the DES Healthcare domain, where he notices the trend going from less than 10 publications per year before 2010 to almost 35 in 2016. The most important healthcare area of research being health and care systems operations. S.

Tavakoli et al. (2008) develop a generic framework for the application of DES in non-manufacturing environments, with a focus on healthcare. They also identify and propose solutions for the weaknesses of "traditional" simulation. They include problems such as time consumption, time dependency, inaccuracy for prediction and cost. In fact recent research into the DES application in non-manufacturing environments has a strong focus on the healthcare sector. (G. Werker et al. - 2009, P. Chemweno - 2014, E. Demir et al - 2017, A. Patel et al. - 2020). S.H. Jacobson et al. (2006) mentions the increasing pressure for quality, rising costs, lower reimbursements and regulations as factors contributing to the use of DES in this sector. The increasingly sophisticated simulation software packages in combination with the new possibilities for simulation optimization facilitate the quick adoption of this technique. The trend of applying DES to healthcare operations is a good example of the evolution of DES as a traditional technique for modelling manufacturing systems to services and hybrid service/production systems like we see at IRI. Healthcare operations is currently the most prominent research environment for this. The factors S.H. Jacobson et al. (2006) mention are also relevant for other systems that do not classify as traditional manufacturing systems.

4.2.3 - Latest developments in the DES domain

Simulation or DES is, especially in the last ten years or so, often used in combination with machine learning or data-driven analytics techniques forming hybrid models. This is done, for example, in support of or provide functions for the simulation model or to test an obtained (intermediate) solution of a machine learning/data-driven analytics algorithm. S. Robinson et al. (2010) distinguish data-driven analytics into descriptive, prescriptive and predictive analytics. F. Peña et al. (2022) use a DES model to represent a digital twin of a mining process, where they use a customised machine-learning model to incorporate the geological variation encountered into the decision making processes. Another application is to use data-mining techniques to analyse simulation output data. P. Martinez & R. Ahmad (2021) show the quantitative analysis of the impact of the introduction of an inspection method scenario in a multi-stage production process. The DES approach allows for a high degree of flexibility in scenario and variability representation. They use a DES model that allows for data driven decision making based on the simulation output. An often seen combination is that of DES and Reinforcement Learning. D.C. Creighton & S. Nahavandi (2002) use a reinforcement learning agent in order to optimise a DES model. Their approach is suitable to optimise complex simulation models or provide robust policies. S. Lang et al. (2020) propose a deep reinforcement learning algorithm in combination with DES for a flexible job shop production problem. They achieve better results than with the benchmark GRASP metaheuristic.

Neural Networks can also be used in combination with DES models. E. Cortes et al. (2020) use Deep Belief Networks to generate a synchronisation scheme for a parallel distributed DES model. C. Wen-Jen & C. Yen-Hsiang (2018) design a patient-centred appointment scheduling application with the use of Artificial Neural Networks and DES.

DES is in its core not an optimization method. DES models can, however, be extended with, for example, heuristics to replicate or imitate elements of analytic stochastic optimization methods, increasing efficiency and practicality of the solution as shown by I. Jackson et al. (2018). This is a part of the Simulation Optimization area of research. They also give a recent argument for the efficiency gain of DES as opposed to a stochastic optimization approach when random components need to be included in the model. M.C. Vélez Gallego et al. (2012) present a simulation-optimization heuristic for configuring a selective pallet rack system where pallets arrive dynamically over time and storage duration, arrival time and pallet height are set to be continuous random variables. G. E. Vieira et al. (2017) propose a meta-heuristic approach for a DES based optimization problem in evaluating the robustness of production schedules in a job shop environment.

Although the inclusion of the techniques mentioned above are not necessary to solve the core problem of this research, the possibility of future additions (either in the machine learning or meta-heuristics domain) makes again a strong argument for the application of DES in this situation.

4.3 - Selection of DES modelling approach for this research based on literature

In this section we will consider the approach for solving the core problem. We explain which method is used, why it is chosen and how it will be implemented. The argumentation makes use of the research domain presented in the literature review and the company analysis of chapter 2.

4.3.1 - Tradeoff (open-source) DES software vs general-purpose programming language

When constructing a DES model to represent a manufacturing process within a company the decision to either use specialised simulation software packages or to build a model from the ground up (with the use of general-purpose programming languages like C++ or Java) must be answered. L. Leemis & S. Park (2004) state that DES model construction is best done with a standard and widely available general-purpose high-level programming language, using the already-familiar tools like the editor, compiler, debugger, etc. The use of special-purpose simulation languages or software/tools can in some scenarios be superior. Especially if the simulation language is already familiar to the developer, if the language or package already exists and is paid for within the company (not relevant for open-source solutions of course) or if there exists the need for a fast development or prototype of the model. Adding to this, since specialised simulation packages often include some kind of animations or visuals their use can be preferred, either for debugging, validation/verification or credibility purposes. In other cases, programming languages are generally preferred (given the skills to use these languages exist of course), since the developer has infinitely more freedom and control in modelling a process, the model can be measured with great precision and realism against the real process, the input and output data type and format can be controlled, the source-code can at any moment be altered if necessary and extra functionality (to either create a hybrid solution, integrate one or more models with the DES model to handle input/output, or simply add GUI or animations) can be added with ease later on in a modular manner. For this research, based on the considerations presented above, the construction of a DES model in a general-purpose programming language is preferred. IRI has no in-house simulation software available, animations are irrelevant in a highly digital process and there exists no problem that requires a prototype or model quickly.

4.3.2 - General-purpose programming languages for DES

When constructing a simulation model in a general-purpose programming language, there are a very high number of different possibilities and approaches. G. Dagkakis & C. Heavy (2016) review in-depth a variety of open-source DES packages, which give an interesting look into the generally preferred programming languages for DES application development. In this analysis it becomes more than clear that there are two languages that are preferred, namely Java with 20 uses out of 48 and C++ with 19 uses. The advantage Java and C++ have against for example Python (on spot number 3) is most likely due to the fact that the first two are compiled languages, while Python is an interpreted language. This gives Python the advantage that it requires less code to achieve the same functionality at the cost of running speed. C++ on the other hand compiles directly into machine code, this has the advantage of increasing running speed drastically. By directly compiling into machine code, the developer can optimise the program for a specific set of hardware (one type of computer), while the program performance suffers when running on other systems (either because less capable or more capable hardware is present, which is not taken into account). Java is another compiled object oriented programming language similar to C++ which has the speed advantage of compiling, but does not directly compile into machine code. This only happens when the program is executed. This means that Java in theory has a slight disadvantage to C++ in terms of speed when it is run on the machine the C++ code is optimised for. Java, however, can take advantage of the capacities of any machine since conversion to machine code only happens after running. This is where the famous Java sentence comes from: “write once, run anywhere”. This corresponds well with the required output of this research defined in 3.1, providing a foundation for a modular and reusable computer program. It can be easily

extended to other platforms and machines when deemed required. R.A. Kilgore, K.J. Healy & G.B. Kleindorfer (1998) confirm these advantages of Java over other languages and in fact state that Java is *the* language by choice for simulation model or component development: "Java will be the foundation of all simulation tools". Although this source is dated, it describes a prediction for the future in the dawn of DES simulation with general-purpose programming languages. Currently, many Java simulation packages are available, confirming the prediction of Kilgore (1998). The availability of packages is a sign of the adoption of DES within the Java community and helps developers in drastically reducing development time. The research of G. Dagkakis & C. Heavy (2016) indeed confirms the prediction of Kilgore (1998) in more detail.

The last advantage Java has over most importantly C++ is the fact that it is significantly more suited to build user friendly applications and modern graphical user-interface (GUI), Another well known example having this functionality is Python. The characteristics of Java however, most importantly the freedom it provides in running efficiently on every system (even in the cloud), the possibility to increase practical usefulness with an intuitive GUI and the general acceptance as language of choice for simulation applications make Java (or languages based on the Java Virtual Machine (JVM) the best general-purpose programming language for DES applications.

4.3.3 - Selection of the modelling approach

For this research and to provide a solution to the core problem a DES model will be developed in the Java computer language. The DES approach is chosen over others because it allows for a representation of a system in great detail, with a good degree of freedom in design without the need of simplifying assumptions that can jeopardise the validity of the model (present is many analytical models). DES is currently the preferred approach for stochastic process modelling, and offers compatibility with a range of (input) probability distributions. We have chosen to develop the model in Java as opposed to an (open-source) simulation package for multiple reasons. The first being that there is no simulation package present within IRI, ruling out commercial simulation packages because of price. Also, choosing a simulation package rather than developing a model in a general-purpose programming language is pertinent when a visual representation of the process flow is essential (i.e. products flowing on an assembly line) or if a model (prototype) is needed in a short amount of time. The fact that the operational process of IRI is rather abstract makes visualising it not very valuable because only little could be shown, also the core problem is an ongoing problem over the long term not requiring an immediate solution. As well as time being available for development of the model, the situation does not require an optimal solution at this moment. If this would ever be necessary, it could be built over the base DES model, although experience with Java, (meta)heuristics and simulation would be required. There are multiple reasons for choosing Java over other languages. Firstly, as shown in the literature review, Java is the preferred language for simulation model development. C++ only falls slightly behind, however Java offers significantly better support for eventual graphical user interface integration in my experience. Another reason for choosing Java is the fact that a compiled Java file is not directly converted to machine code. This is the case with C++ and offers therefore the possibility to optimise a program for one specific machine. We, however, do not intend for the model to be able to run efficiently on only one machine, but eventually be used by multiple people for analysis (this is probably beyond the scope of this research however). Java will convert to machine code based on the machine it runs on, following the famous phrase "code once, run anywhere". Lastly, although similar in design and functionality, my skill is more developed in Java rather than C++. As a general guideline and reference the book of J. m. Garrido (2012) Object-Oriented Discrete-Event Simulation with Java: A Practical Introduction will be used.

5 - IN-DEPTH ANALYSIS OF THE OPERATIONAL PROCESS

Chapter 2 already included a description of some operational processes related to the selection of the core problem. This chapter will expand further on these processes. It will also provide an overview of the total operational process. The divisions of the operational process with interest for this research will be converted into a queueing network, as a stepping stone for eventual implementation in a DES computer application.

5.1 - BPM of the total operational process

Figure 7 on the next page presents the total operational process of IRI in the form of a Business Process Model. The BPM model is constructed by spending a day with every operational division. Activities have been put in logical order and afterwards the BPM was verified with the process expert of each division. For the divisions Measurement Science and Delivery there is no process structure added to the activities, this is because each of these two divisions consist of one person performing the defined activities in an ad-hoc fashion.

5.2 - Stochastic parameters in the operational process of IRI

All real life systems encounter some form of stochasticity. When modelling a system they can be included or left out of the model. In the case of IRI, we are interested in what effect the stochastic aspects of the process have on its performance. The factors of variation present in the operational process of IRI have been first mentioned in subsection 2.2.2 and are essential to include in the model to gain insight in their effect and possible modifications to the process that could mitigate their negative effects. In this section we will further elaborate on these. At least three important factors of variation in the operational process can be identified:

- The first is related to the (input) workload variation (i.e. the variation that each department experiences within the number of jobs that have to be completed). This variation differs per department and is multi-layered (i.e. the number of incoming EANs is uncertain, but also in which category they will belong and how many attributes belong to this category is variable) and is largely outside the sphere of influence of the company. This is the case because IRI buys this data (either with money or with their offerings such as databases and reports), and wants as much as possible, including all new product introductions (from for example Jumbo).
- The second is a strong variation in processing times, which can also involve waiting time and rework. The main source of the variation is the fact the processing is done manually by multiple people. An earlier identified problem regarding high employee turnover is a strong influencer of this variation as well (as new employees need time to get up to speed).
- The third is the effect of new retailer integrations. These retailer integrations will cause an overflow workload on the operations department and involve mainly two factors of uncertainty namely: an uncertain exact arrival of this overflow workload and an uncertainty in the number of jobs this overflow involves.

Many of the problems identified in Section 2.1 can be associated with the factors of variation introduced above, making it necessary to model the operational processes within IRI related to unknown EAN processing as a stochastic system; we will therefore disregard techniques not suitable to model stochastic systems in the rest of this literature review.

THIS FIGURE CONTAINS SENSITIVE INFORMATION AND IS LEFT OUT FOR CONFIDENTIALITY PURPOSES - CONTACT IRI AT DIRK.ROOST@IRIWORLDWIDE.COM IF YOU WISH TO GAIN ACCESS TO THIS FIGURE

Figure 7 - BPM of the total operational process of IRI

5.3 - The distinction between production and services industries

IRI is an interesting example of a hybrid company that can be seen as both a (data) production company and a consultancy agency. All consultancy centred activities depend on the output of the operations division that operates in intermittent production cycles. Because of this, definitions of both a production as a service environment is needed to make sure the process of IRI is analysed in the right way. The application of many Operations Research techniques such as simulation, in particular DES, Linear/Stochastic Programming, Queueing Theory and many more have mainly evolved out of traditional production/manufacturing research and applications. Of course, the (first) industrial revolution can be seen as the historic start of the research into the optimization of production processes. Only in the last few decades or so, traditional industrialised countries have moved to a mainly services oriented economy. Many big production companies have moved their production plant(s) to low wage countries. This has initiated a strong interest within the Operations Research domain to more extensively analyse the services industry. S. Robinson (2005) gives examples of some service industries of interest for Operations Research such as: airports, call centres, fast food restaurants and healthcare (which has seen significantly increasing research interest in the last fifteen to twenty years - shortly mentioned in subsection 4.2.2). S. Robinson (2015) makes an important distinction between a production and a services centred process, which is that in the modelling of service systems *customers* are present (who in a sense co-produce the product), while in a manufacturing environment this is not the case. As this definition does not correspond with the case of IRI, we will not dive further into this area and use the general definition of a production environment that corresponds with the intermittent production cycles of the Operations division of IRI: a physical facility that realises the process of transformation of a variety of inputs into outputs.

W. Perales (2001) provides a general definition for a manufacturing process: manufacturing at its terms is the process of converting raw materials to finished products by using several processes, machinery and energy. This definition is in accordance with the operational process of IRI, where the raw materials are considered to be the arriving unknown barcodes, the final products are the databases with classified EANs per product category which are used by the consultancy side of the company for analysis. The processes that contribute to the conversion of the raw materials into the finished products can be seen in Figure 7. The machinery in this manufacturing process are the employees per operations division, therefore the definition of W. Perales (2001) could be further generalised by changing machinery to "processors". To make a further distinction in the manufacturing process of IRI we use the following Figure of W. Perales (2001) that categorises manufacturing systems into 4 different types:

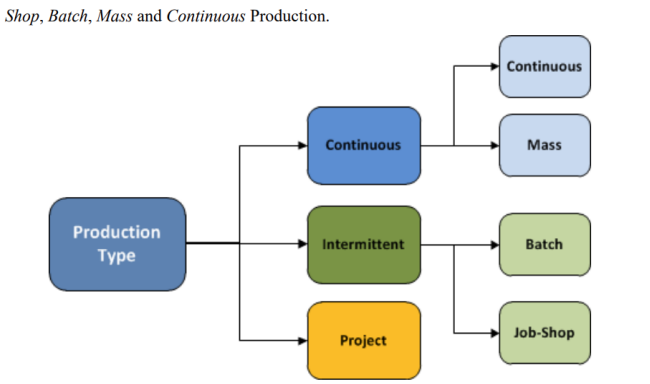


Figure 8 - Diagram for the different types of manufacturing systems and their classifications according to the production type (souce: W. Perales (2001))

The operational process of IRI consists of separate production periods of 4 weeks (so not continuous and not project based), where a multitude of data deliveries (unknown barcodes) are processed by the system. The type of manufacturing system therefore most closely resembles an intermittent batch manufacturing system as can be derived from Figure 8. The arriving unknown barcodes arrive in batches and are all considered the same, before they are assigned to a product category, which implies low variation. A job-shop classification implies a high product variation, which is considered when setting up a job-shop process. Within the IRI operational process a job-shop setup could only be used after assignment to product categories, because only then enough information is available to set-up a process considering product variation.

5.4 - Selection of the research focus area

The complete operational process of IRI is quite complex, consisting of multiple interrelated departments performing a great variety of different tasks (not only directly related to data production). This fact makes it necessary to decrease the focus area of this research to a subsection of the operational process, to both allow for a more meaningful research result and to make sure the scope does not get out of hand. As this is the first research into the performance of the operational process, it makes sense to set the focus area to the part of the process that has the highest direct relation towards data production output of the process. It is the part of the operational process most commonly referred to as the bottleneck. Chapter 2 introduced the Coding and DBA departments as those steps in the operational process responsible for the processing of new EAN arrivals. These two departments will also be the focus area of the process for analysis with the use of the DES model. This means that both the Data Input, the Field (Inside & Outside) and the Global Operating Centre (GOC - team in India) parts of the process coming before the Coding and DBA processes will be disregarded. Although some inefficiencies have been identified in these processes, these departments are expected to be less affected by an increase in workload then Coding and DBA. Also, the number of delays experienced from internal problems/inefficiencies within these departments in the rest of the company is significantly less to almost nonexistent. This means the analysis of this research assumes that all necessary information (data input) is already present in the Pangea system used by Coding, and that the DES model starts with generating a EAN arrival amount based on a fitted distribution from internal IRI data (performed in chapter 6).

After the DBA process, there exist two departments handling the follow-up processes as can be seen in Figure 7: the Measurement Science department and the Delivery department. Also here it is true some inefficiencies exist within the current process, however analysis here is significantly more difficult and has less expected added value. As shortly mentioned in Section 5.1, at the time of writing both of these departments consist of only one person performing mostly ad hoc activities without a clear process structure in place. This means that the analysis of this research and the DES model will end at the last stage of the DBA process, which is the submission of a database with newly processed EAN's to the IRI mainframe.

5.5 - Queueing system of the Coding & DBA process

THIS FIGURE CONTAINS SENSITIVE INFORMATION AND IS LEFT OUT FOR CONFIDENTIALITY PURPOSES - CONTACT IRI AT DIRK.ROOST@IRIWORLDWIDE.COM IF YOU WISH TO GAIN ACCESS TO THIS FIGURE

Figure 9 - Queueing system of Coding & DBA process

As a starting point for the DES model development, we have converted the BPM model of Section 5.1 to a queueing system representation of the Coding and DBA process presented in Figure 9 on the next page. This queueing system will be used as reference material to construct a DES model which corresponds as closely as possible to the actual process without sacrificing model performance (in modelling elements that do not directly contribute to the defined performance indicators of this research).

5.6 - Rework modelling

Rework is an aspect of the operational process which is often mentioned as a part of the tasks of divisions in the operational processes. We include this aspect of the process into the model to remain as close to reality as possible. Until now we have considered solely the workload that arises from incoming data (at the start of the process). Besides this workload however, another factor contributing to workload in the Coding and DBA departments directly related to EAN processing is rework. Rework is the term for all jobs that have been processed before by one or more departments, but at a certain stage in the process it is determined that one or more parts have not been done correctly and must be redone. There exist two forms of rework that are relevant for the workload of the Coding and DBA departments:

- Internal requests (coming from inside the Coding & DBA departments when when a mistake is identified or a change to an ongoing job is needed)
- External requests (coming from outside the Coding & DBA departments, either from another department in the company or from a client)

As the combination of EAN arrivals and rework makes out the workload of these two departments related to data production, rework will be included in the DES model. Although we know that rework exists in the process from the performed process analysis, we do not yet know the size of the added workload due to rework or the effect this has on performance of the process. In Section 6.3 and 6.7 the amount of workload related to rework in the current process will be extracted from internal process data. Subsection 7.6.3 expands on how rework is integrated in the DES model during development.

5.7 - Chapter conclusion

We have analysed the complete operational process and converted it to a BPM. Next, the stochastic parameters this process copes with and that are necessary to incorporate in the DES model have been further elaborated. The most important stochastic parameters of the process are the UPC arrival amount, the variation in processing times and the variation that arises with a new retailer integration. Next, the distinction between services and manufacturing industries is made. As IRI could be considered as both it is important to provide a distinction between the two, to make sure the final model models the process in the right way. Based on the analysis of the process, the operational processes of IRI can be considered as a manufacturing process with human resources. Based on Sections 5.1, 5.2 and 5.3 the focus area of the model is determined, which is the combination of the Coding and DBA departments. These are by far the largest divisions in the Operation department and perform the key tasks to get from raw data record to database that can be sold and used. Good functioning of these departments is vital for all other departments. Additionally, these two departments are first named as the bottleneck of the company as problems occur often (delays, overwork, mistakes) which, in turn, affect the whole company. Lastly, two elements are derived from the BPM which will serve as direct input for the implementation of the DES model namely: the queueing process representation on which the DES application will be based and the way rework appears and is handled within IRI.

6 - DATA COLLECTION & ANALYSIS

This chapter will include a list of requested internal data points, the eventual available data-sets and an analysis of the data to provide some input for the DES model. For the analysis of the data the Python programming language is used.

6.1 - Data collection

As input for the simulation computer program data on the performance and characteristics of the current process is used, to make sure the model will represent the actual system (processes) as accurately as possible. CSV datasets extracted from one of the systems IRI uses named Pangea over a period from 2018 until now are used for analysis. The data necessary as input for the DES model per department are presented below:

Coding department

- Number of employees (NL)
- Weekly unknown EANs entering system
- Percentage automatic classification (both Keycatting and assigning attributes)
- Number of attributes per Keycat
- Distribution of categories of processed EANs
- Processing times on individual jobs (Keycatting, Coding individual attributes (incl. internet search, hint file check)
- Nr. jobs completed per period per employee
- Rework percentages

DBA department

- Number of employees (2 different DBA divisions)
- Number of jobs completed per period (per division)
- Number of new adds per period
- Distribution of new adds over databases (which databases receive more new adds in general and which don't)
- Time spent on individual jobs (checking category/rulebook, Placement, nil reporting, client feedback, database uploads)
- Nr. jobs completed per period per employee

Datasets used

Four internal databases were used for analysis to extract useful information as input for the model:

- UPC Arrivals (data over 2018-now)
- Keycat Attributes & Recipe (data until now)
- NL Coding Timings
- DBA metrics dashboard database

6.2 - Unavailable data & solutions

Not all necessary data for the DES model was available or measured. To avoid complications that could arise with our own measurement of needed data, we have chosen to use known data and process specialists within the company to make informed estimations on missing data. For example, the DBA Placement process has a lot in common with the Keycat and Coding process (some years ago it was part of the same process), so it is assumed that the fitted distribution on the Coding processing times also applies for the Placement processing time, with a different mean value. Also it is assumed

that all barcodes that are coded, need to be processed by Placement, so no data on actual Placement numbers is needed. Assumptions have been verified by the management of the specific divisions to make sure that no unacceptable simplifications are made. The substantiation of the assumed distribution parameters can be found in subsection 6.6.2 and 6.6.3.

6.3 - Data analysis baseline - UPC Arrival & Keycat Attributes datasets

In this section the EAN Arrivals (called UPC (Universal Product Code) in the databases) and Keycat Attribute datasets will be analysed (with the use of Python) to extract useful input for the DES model.

6.3.1 - Weekly UPC Arrivals (2018 - now)

Figure 10a and 10b show the weekly UPC (unknown barcodes) arrivals both in the total span of the data set and excluding the effect of a new retailer introduction (Bol.com) that started in December 2021. We can identify one outlier in September 2020. After consulting with the company's data expert, we will disregard it in the analysis as something happened which does not resemble normal workload circumstances well.

THIS FIGURE CONTAINS SENSITIVE INFORMATION AND IS LEFT OUT FOR CONFIDENTIALITY PURPOSES - CONTACT IRI AT DIRK.ROOST@IRIWORLDWIDE.COM IF YOU WISH TO GAIN ACCESS TO THIS FIGURE

Figure 10a - weekly UPC arrivals 2018-2022

Figure 10b - weekly UPC arrivals 2018-2021

6.3.2 - Weekly UPC Arrival Distribution

As input for the DES model we have used a probability distribution function fitted on the occurrence. Figures 11 and 12 on the next page show plots of the empirical data of weekly UPC arrivals. The Distfit python library is used to determine the best fitting theoretical distributions. Distfit fits 89 distributions and shows the best fitting distributions based on the residual sum of squares (RSS) measure. This is a measure of the discrepancy between the data and an estimation model and is commonly used in parameter and model selection. The best fitting distribution that is available in the Java Simulation Library (JSL - see 7.2 for introduction and substantiation of its use) are also shown in Figures 11 and 12. For the period 2018 - 2019 the normal distribution has the lowest RSS value which is 1.07398e-06 For the period 2018 - 2020 (excluding September and December) the lognormal distribution has the best fit with an RSS score of 4.74138e-07. As we are essentially just interested in finding the best fitting theoretical distribution function for the data, the exact RSS value is not relevant. It is only necessary that we evaluate the fit of each distribution with the same statistic and select the distribution with the lowest value. If the empirical data would not have resembled a theoretical distribution, we could have used the empirical data directly using an empirical distribution. Since Figure 11 and 12 show that the empirical data does resemble the best fitting theoretical distribution and the fact that we do not possess a lot of data points (especially for the period 2018-2019), since we are looking at weekly data, a theoretical distribution will be used.

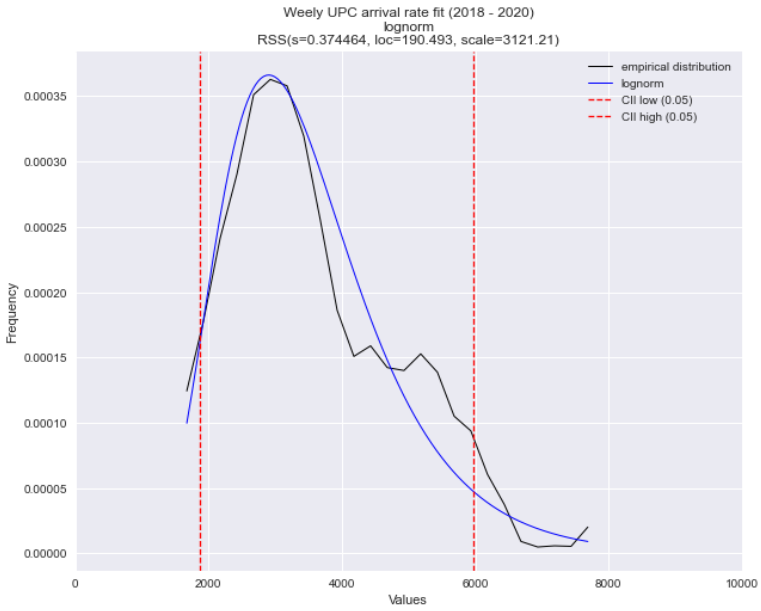


Figure 11 - Histogram of arrivals with fitted lognormal distribution 2018 - 2020 (excl. Sep & Dec 2020)

Figure 12 - Histogram of arrivals with fitted normal distribution 2018 - 2019

6.3.3 - Workload percentages

In the DES model a initial UPC arrival amount will be generated according to the distribution fitted above. From there percentages based on historical data are used to determine where, when and if these UPCs are processed. Table 2, 3, and 4 present the percentages extracted from the UPC arrivals database for the Keycatting stage, the Coding stage and related to rework.

THE FOLLOWING TABLES CONTAIN SENSITIVE INFORMATION AND ARE LEFT OUT FOR CONFIDENTIALITY PURPOSES - CONTACT IRI AT DIRK.ROOST@IRIWORLDWIDE.COM IF YOU WISH TO GAIN ACCESS TO THESE TABLES

Table 2 - Workload percentages at Keycatting stage
Table 3 - Workload percentages at Coding stage
Table 4 - Rework percentages at Keycatting & Coding stages

In the Tables 2, 3 and 4 the years 2021 and 2022 are already left out to get a good sense of the baseline workload as the peaks that can be seen are solely the effect of the Bol.com integration. What can be seen in the tables above is that, while the years 2018 and 2019 closely resemble each other, 2020 deviates significantly even excluding outlier months September and December. This is why we will use the years 2018 and 2019 to represent the baseline workload scenario as input for the model. A workload as seen in 2020 could be used to represent a slightly less predictable scenario.

6.4 - Data analysis overflow workload - UPC Arrival & Keycat Attributes datasets

This section will analyse the same datasets again to gain insight into overflow workload conditions based on data of the Bol.com integration (starting at the end of 2020). It will provide a foundation for the process alternative of Section 8.11. For this analysis we have used a Python forecasting function using three different forecasting models which will be introduced now to give the reader a sense of their workings, as all three are based on a different method of forecasting.

6.4.1 - Forecasting function models

To estimate the effect a new retailer integration has on the UPC arrival amount, we have used a forecasting function based on three different forecasting models (Exponential Smoothing, Facebook Prophet & Neural Prophet) to forecast the UPC arrivals for the period December 2020 to april 2022 based on the period 2018 to november 2020. The forecasting function will provide fitted values on the input data and the final forecast based on the model that has the lowest mean average error between fitted and real values. This will give an impression of the baseline UPC arrival amount as if the Bol.com integration never happened. The three different forecasting models used here serve as tools to be able to separate base workload from overflow workload. The three models are applied in their basic form and are in that sense not part of the DES approach to the research and have also not been part of the literature chapter. Most importantly the Neural Prophet model, as its Neural Network properties will be used to gain a more precise idea about the expected effect of a retailer introduction. Since this is based on only 1 new retailer introduction and we are only looking at one variable (UPC arrivals) this analysis can be further extended which will be indicated in the recommendations.

Exponential Smoothing is a classic time series forecasting method (first suggested by Brown (1956)) that uses weighted averages of past observations with weights decaying exponentially as observations get further back in time (Hyndman - 2013). Different forms exist of Exponential Smoothing:

- Simple: no trend, or seasonality is considered
- Double: trend is considered as a parameter that is constantly updated
- Triple: both trend as seasonality is considered. The seasonality parameters can be considered for repeating periods such as months, weeks, days and are updated when the same period occurs again.

In the applied forecasting function all methods are considered and parameters are optimised to gain the best forecast within exponential smoothing.

Prophet is a modern open-source algorithm from Facebook/Meta that is meant as a complete forecasting solution for time series data to be used in the programming languages Python and R. Prophet works with the approximation of three functions that combined provide the (fitted) forecast function. The functions Prophet uses are as follows:

- The growth function, which is a function approximation of the trend of the data (linear, logistic or flat)
- The seasonality function, which is an approximation of a function based on a Fourier series as a function of time
- The holiday function, which is able to change the forecast based on default or user defined dates

A combination of these three functions is used in an additive manner to arrive at the fitted approximation of the function of the data and the forecast. Additionally, Prophet is able to handle missing data points and outliers.

Neural Prophet is a hybrid extension of the Prophet model by combining the classic time series forecasting with deep learning methods. Neural Prophet adds three extra functions (or modules) which are (Triebe - 2021):

- Regression effects at time t for future-known exogenous variables
- Auto-regression effects at time t based on past observations
- Regression effects at time t for lagged observations of exogenous variables

As a result, Neural Prophet generally achieves higher accuracy with less predictable data, or data with less clear trend or seasonality.

6.4.2 - Overflow workload analysis

As can be seen from Figure 10a (and 13) in December 2020, the first Bol.com data arrived, meaning a high number of new unknown UPC arrivals. We can see that this retailer integration first contributed to a very strong increase (overflow) in UPC arrivals (~ Dec. 2020 - June 2021) and then increased the average and standard deviation of the new baseline scenario (from Jul. 2021). For the scenario analysis of a retailer integration using the DES model we will separate two cases shown below, we will use the forecasting models to look at these scenarios in the data from a variety of different viewpoints.

- The overflow UPC arrival amount present in the first few months after a retailer introduction (this is equal to all of the products of such a retailer which are not yet known and classified by IRI)
- The increase of the mean and the standard deviation of the baseline UPC arrival amount after the processing of the overflow arrival amount (this is the previous baseline arrival amount, plus the new product introductions of

the newly integrated retailer).

THIS FIGURE CONTAINS SENSITIVE INFORMATION AND IS LEFT OUT FOR CONFIDENTIALITY PURPOSES - CONTACT IRI AT DIRK.ROOST@IRIWORLDWIDE.COM IF YOU WISH TO GAIN ACCESS TO THIS FIGURE

Figure 13 - Real UPC arrivals (blue) and estimated UPC arrivals without retailer integration (red)

Compared to the baseline UPC arrival amount (2018-2019), the UPC arrival amount during and after a new retailer introduction has a clear effect on both the mean and the standard deviation of the UPC arrivals. We make two forecasts with the Neural Prophet model based on different input data to compare the expected arrival rate. We compare the forecasts until April 2022 (combined with the fitted values) of the Neural Prophet model when only using input data over 2018 and 2019, and when using input data over 2018 until 2021 (so including the integration overflow). The results can be seen in Figure 14 on the next page.

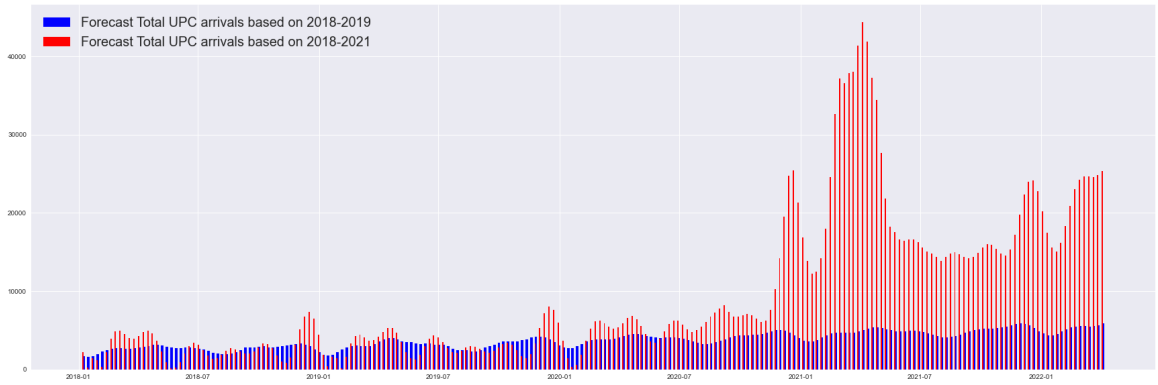


Figure 14 - Neural Prophet forecasts with input 2018-2019 and 2018-2021

As can be seen in Figure 14, the addition of the Bol.com integration data dramatically increases (red in Figure 14) the forecasted/expected total UPC arrivals compared to the estimated baseline arrival amount (blue in Figure 14). However, not *all* UPC arrivals directly affect the workload, since only those UPC's that exceed a defined threshold (based on sales volume) are eligible for a Keycat assignment (and should therefore be assigned to a Keycat). Details could already be seen in the workload percentages tables of subsection 6.3.3, This is especially relevant to realise in the scenario of a new retailer introduction as can be seen in Figure 15. Figure 15 shows the weekly UPC arrivals that have exceeded the eligibility threshold and an estimation of the arrival amount of eligible UPC's without the Bol.com integration based on the forecasting function. The difference between the real eligible UPC arrivals and the forecast is dramatically smaller. This is most likely because the larger an assortment of a retailer is, the more products do not exceed the total sales level necessary for IRI to process the product. This is in line with the Pareto principle where generally 20% of products generate 80% of revenue. However, the further an assortment increases, the flatter the slope of the last 80% becomes. In a product category products with the highest sales level are processed first, because they contribute the most to the total coverage of the product category based on total product sales.

THIS FIGURE CONTAINS SENSITIVE INFORMATION AND IS LEFT OUT FOR CONFIDENTIALITY PURPOSES - CONTACT IRI AT DIRK.ROOST@IRIWORLDWIDE.COM IF YOU WISH TO GAIN ACCESS TO THIS FIGURE

Figure 15 - Real eligible UPC arrivals (blue) & estimated eligible arrivals without retailer integration (red)

The difference between total UPC arrivals and eligible UPC arrivals becomes even more clear when comparing the Neural Prophet forecast until April 2022 (using input data from 2018 until 2021) of the total UPC arrivals and of the eligible UPC arrivals. This gives an estimation of the expected UPC arrivals based on available data which is shown in Figure 16 on the next page.

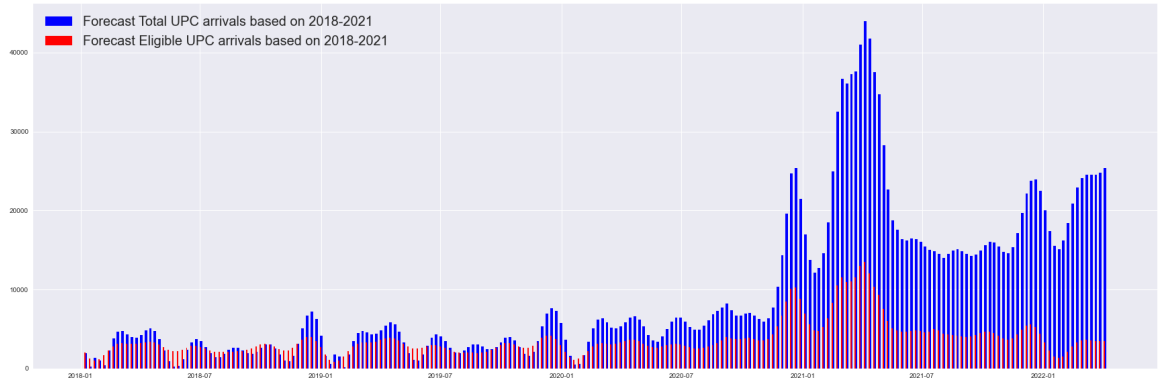


Figure 16 - Neural Prophet forecasts on total and eligible UPC arrivals

As can be seen in Figure 16, the expectation of total UPC arrivals (blue in Figure 16) has a stronger variation than eligible UPC arrivals (red in Figure 16). Also a trend towards a higher percentage of ineligible UPC arrivals can be seen as the total UPC arrival amount increases. This intertwines with the observation that while a relatively strong positive trend in total UPC arrivals can be seen, a positive trend in eligible UPC arrivals (aside from the temporary peak in 2021) is less clear, or at least significantly less strong. When comparing Neural Prophet output comparing model fit based on data including and excluding the Bol.com data, a clear increase can be seen in expected variation of eligible UPC arrivals and an increase in the expected mean after the initial peak in the beginning of 2021. The peak in the beginning of 2021 alone could cause the model to increase the variation over the whole prediction, however based on this data alone (data on only one retailer integration this is difficult to verify. Either data on more than one retailer integration is needed or deeper research into characteristics and predictors of the effect of a retailer integration on eligible UPC arrivals must be done. What we can see however is that the model forecast of the eligible UPC arrivals expects quite drastically lower variation and a drastically lower mean after the initial overflow workload..

Figure 17 shows the expected number UPC arrivals that are eligible remain fairly constant with an increase in total UPC arrivals. We show this using data over 2018 until September 2020 to predict the total and eligible UPC arrivals. The average percentage of total UPC arrivals being eligible (based on forecasted values) in period 2018 until 2019 is 92% and in the period 2020 until may 2022 57%, with an increase in total UPC arrivals of 75%. To realistically model an increase in UPC arrivals, we need to make sure we decrease the percentage that is eligible.

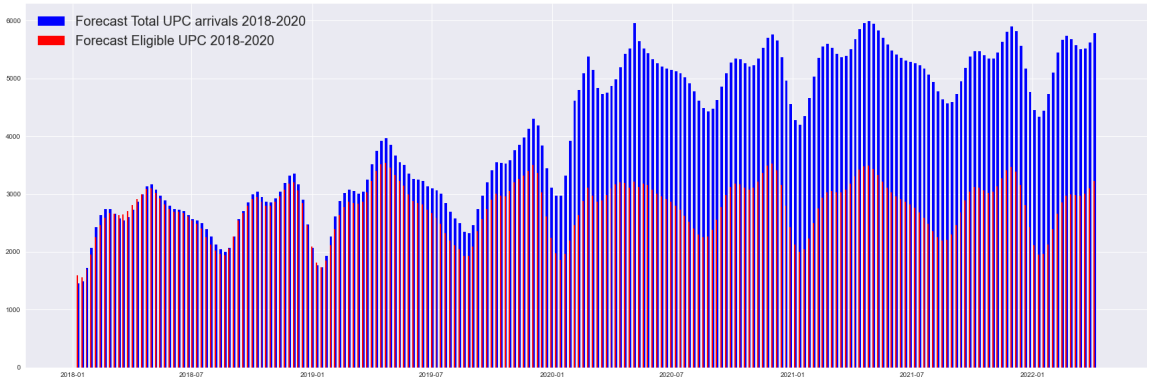


Figure 17 - Neural Prophet forecast of eligible and total UPC arrivals

Figure 18 confirms the statement from above where the fitted exponential trend line of eligible UPC arrivals increases notably less than the trendline of total UPC arrivals.

THIS FIGURE CONTAINS SENSITIVE INFORMATION AND IS LEFT OUT FOR CONFIDENTIALITY PURPOSES - CONTACT IRI AT DIRK.ROOST@IRIWORLDWIDE.COM IF YOU WISH TO GAIN ACCESS TO THIS FIGURE

Figure 18 - Increase eligible UPC arrivals with increasing total UPC arrivals

If we only look at the Neural Prophet expectation of the eligible UPC arrival amount with and without the introduction of Bol.com in the training data as shown in Figure 19, we can see a stronger deviation in the period before the introduction which could indicate an expected increase in deviation when a new retailer is introduced. Also, we note that after the introduction workload peaks (first half of 2021) are smoothed out, the Neural Prophet prediction shows a higher mean than when the introduction is left out of the training set.

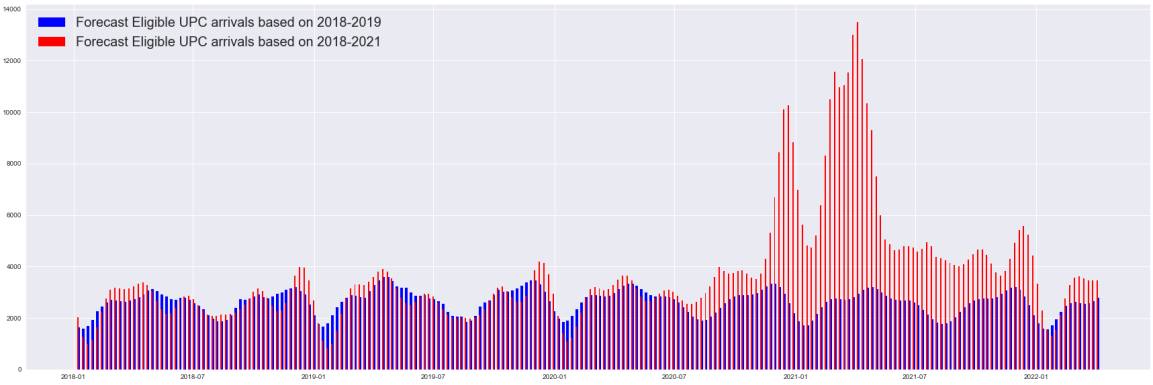


Figure 19 - Neural Prophet forecast of eligible UPC arrivals (input 2018-2019 & 2018-2021)

In this section we have analysed UPC arrivals in the case of a new retailer introduction or overflow workload. From these analysis it can be concluded that both for total and eligible UPC arrivals the mean and the standard deviation of the new

base workload will increase (after the initial workload peak). We have also shown that the amount of eligible UPC arrivals does not increase linearly with the total UPC arrivals. Smaller retailer introductions will therefore have a higher percentage of eligible UPC arrivals than larger retailer introductions.

6.5 - Product category (Keycat) occurrences

This section will look at the Keycat occurrences in the data over the period 2018 - 2019. As we know, arriving UPC are codes belonging to certain products. These products belong to a category (set of similar products defined by IRI), which is assigned at the Keycat stage. Different Keycats have different properties such as number of attributes and number of databases the Keycat is part of which affects workload. We construct a distribution function that we can use in the DES model.

6.5.1 - Cumulative Distribution Function of Keycat occurrence

To represent a realistic occurrence of UPC Keycats in the model and differentiate between Keycats that occur more and less often in the real world, a cumulative distribution function is created of the occurrence of Keycat types in the data over 2018 - 2019. This way, a Keycat that occurs more often in the data also has a higher chance of being generated by the model. A visual representation of maintained Keycats (Keycats that need to be processed further than only assigning a Keycat) for the Keycat that occurs the most in the data to the one that occurs the least is presented in Figure 20. A uniform distribution is used to generate a percentage which is then reflected on the inputted cumulative distribution.

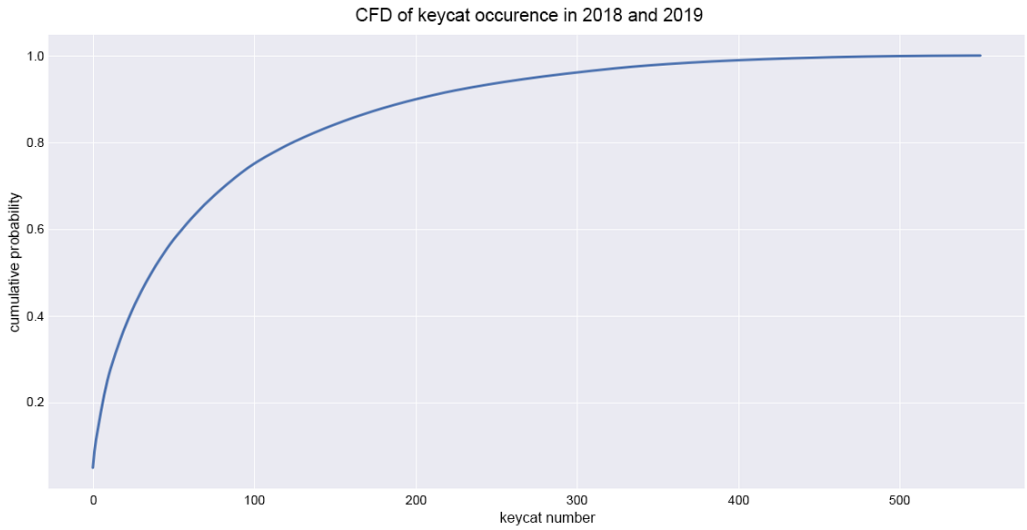


Figure 20 - CDF of Keycat occurrence

6.5.2 - Maintained Keycats

Not every UPC that arrives via the Data Input station in the IT system of IRI will eventually go through every phase of the operational process. At IRI a distinction is made between maintained and non maintained Keycats. Non maintained Keycats will not be coded since they are considered as having no direct added value for IRI to process further, mainly because there exist no paying clients within the Keycat category. A list of maintained Keycats is extracted from the dataset and put into a .csv file. This list of maintained Keycats is used in the simulation model to determine if an UPC which arrived and was Keycatted will continue in the system to the Coding part of the process or go directly to the dispose object and leave the simulation.

Now we have extracted all necessary data from the UPC Arrival & Keycat Attributes databases we move on to the next database to gain insight in the processing times of certain tasks within the process.

6.6 - Processing times

In this section we analyse historical internal data to gain insight into the realised processing times per task of the operational process. The Coding Timings database is used to analyse task processing times that have occurred per action at IRI. The best fitting theoretical distribution is then used in the DES model. This distribution is determined by using the Python Distfit library, where we only check the theoretical distributions available in the Java Simulation Library.

6.6.1 - Processing times of the Coding process (Keycatting & attribute Coding tasks)

First we determine the distribution of the task times for the Keycatting task. All timings have been included where the data record starts with the state "new movement", which means a new incoming UPC, and ends with "Keycat approved". These records show timings for a complete Keycatting action. The lognormal distribution is the distribution with the best fit with an RSS score of 0.00238356. The distribution of the empirical input data and respective lognormal fit are shown below in Figure 21.

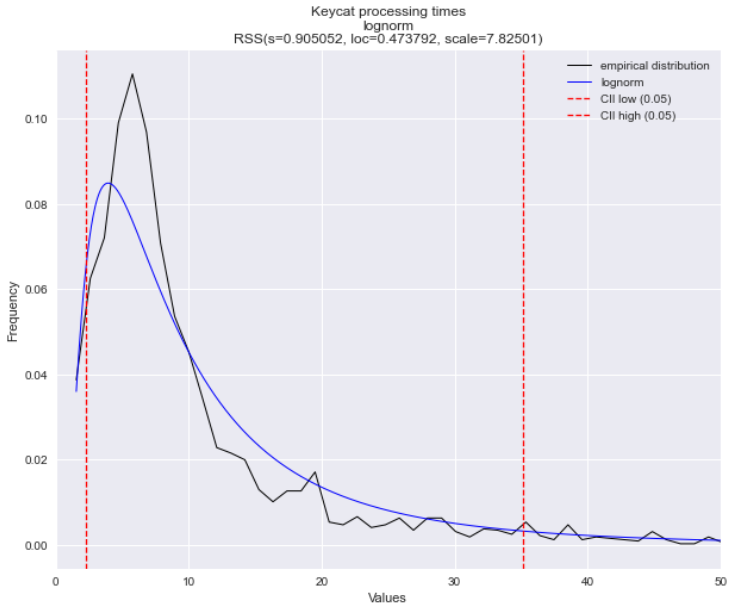


Figure 21 - Histogram of Keycat processing times & fitted lognormal distribution fit (mean: 12.25, variance: 176.20)

To determine the Coding task times, we use data records in the database that have a first state of "Keycat approved" and an end state of "completed". These records signify a Coding action. However, the time indication for these records is for the Coding of *all* attributes of a certain Keycat. We are only interested in the task times of 1 attribute Coding for the simulation model since they will be multiplied by the number of attributes of a Keycat in the simulation model. This means that here we have to divide the time indication by the number of attributes of the respective Keycat of the data record. Again the lognormal distribution shows the best fit with an RSS score of 0.000102427. The graph of the empirical distribution including the fitted distribution is shown in Figure 22 on the next page.

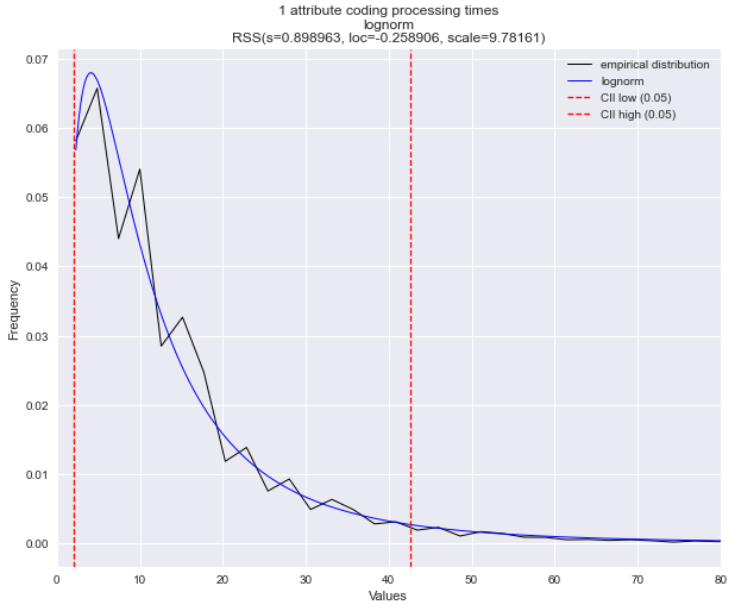


Figure 22 - Histogram of 1 attribute Coding processing times & fitted lognormal distribution (mean: 14.39, variance: 267.00)

6.6.2 - Processing times of the DBA process

There is no measured data on the DBA part of the process. To obtain distributions for these tasks we can either measure the task times or make informed assumptions. As mentioned before, the DBA process has many of the same characteristics as the Coding process, therefore we will make assumptions on the distribution and the parameters of the input distribution for the tasks with missing data. The distributions for the following tasks will be obtained: Placement, Placement validation, NIL reporting, feedback processing and database submission. The accuracy of the results of the simulation model can be increased by accurate measurement of these tasks in the real process. However, with the goals of this research in mind, making informed assumptions saves a lot of time without a significant expected loss of strategic insight into the process.

The Placement task is very similar to the 1 attribute Coding task. Instead of an internet search for an attribute, the employee needs to consult a hint file. However, it is a task which can be done faster than Coding. We therefore assume that the Placement distribution is comparable to the Keycatting action distribution. The Placement validation task checks a STUB file on whether the Placement action has been performed with success. This takes significantly less time than a Placement action, so we assume the same distribution with a decreased mean and variance (mean: 4.08, variance: 58.73). The NIL reporting action consists of making a New Item List of a category, and sending the NIL of every category a client pays for to the client to be checked. Making a NIL is a one click action and then a mail has to be sent with the right NILs. We assume this action to be comparable to the Keycatting action with a slightly higher mean (mean: 15.31). The feedback processing consists of replacing the new adds that come back with feedback from the client. This action is the same as the Placement action, so we assume the 1 attribute Coding action distribution. The database submission action is the submission of a STUB file with the new adds to the mainframe. This action is assumed to be the same for every submission, so the standard deviation is expected not to be anything else than it being a human action (other actions are also influenced by other factors such as category difficulty, which is influenced by things such as type of product, length of the hint file and agreements with the client). For this action we assume the Keycatting distribution with a significantly lower variance (variance: 35.24).

6.6.3 - Processing times of employees in training

Employees in training can be added to the simulation model to get an idea what kind of impact newly attracted employees without experience can have on the performance of the process. We assume that an employee without training and experience performs worse than fully trained employees with experience. How much this performance gap is, however, is more difficult. For simplicity we assume that an employee in training performs a set percentage worse than a full employee. After consultation with a DBA process expert within the company we have set this percentage at 60% (so they are 60% less effective than full employees). We create a new distribution for these task times by multiplying the fitted mean and variance parameters by 1/(1-0.6) = 2.5. An example of the modification of the distribution between full employee and employee in training is shown in Figures 23a and 23b.

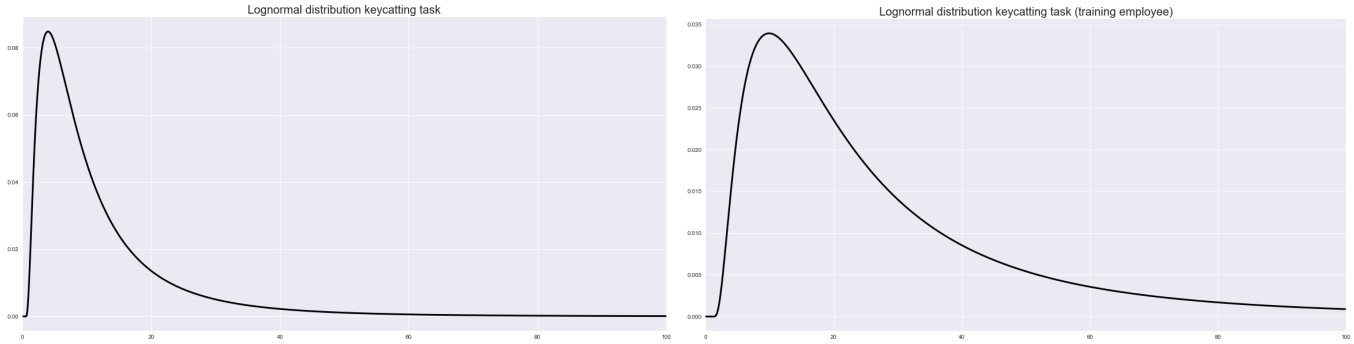


Figure 23a & 23b - Lognormal distribution of Keycatting task, full employee vs employee in training

While this is a quite simple implementation of this aspect of the process, more complete implementations will need additional research. Some areas of this aspect of the simulation that can be improved are:

- New employees need to be trained by 1 or more existing employees, reducing the effectiveness of these employees.
- An employee in training will gradually get better, this will however range outside of the period of analysis chosen for this research (1 production period, 4 weeks).
- A more in depth analysis of the probability distribution of task times of employees in training will give a more realistic view of processing times of these employees than just multiplying the task times.

We expect that research into the points above can be quite time consuming (measurement of human behaviour and the mapping of evolution of processing times for example). The integration of the information found in this additional research can be "easily" included in the DES model of this research. It should be nothing more than the modification of certain random variables and minor process adjustments.

6.7 - Client feedback

At the DBA station, generated lists of new items per category are sent to the client for a final check after the items are placed in a Keycat file. To get an idea of the chance of feedback on an average DBA new adds in a category the DBA metrics dashboard database file is used. Data in this database ranges from 2019 to 2022. Over this period the total number of new adds was 523181. In the NIL file (New Introductions Listing) a client can have feedback on the Placement of a new add. This feedback gets sent back to IRI and causes feedback rework. Over the period available in the database the total number of new adds with feedback is 11060. This gives a chance of 2.1% a new add receives feedback. If we look at a more representative period for a baseline performance analysis such as 2019 - 2020 we get a chance of 1.9% a new add receives

feedback. In the simulation model we will use a percentage of 2% to represent the number of new adds that get back with client feedback. The time before feedback is received is generally around 24 hours. No data on the exact time is yet available. For this research the time before feedback is received back is assumed to be normally distributed with a mean of 24 hours and a standard deviation of 5% of the mean. It is generally the case that the client has one day to provide feedback if necessary. To improve the accuracy of the simulation model this can of course be measured and put into the model like the previous data.

6.8 - Chapter conclusion

In chapter 6 all relevant available data is analysed and distilled into usable input data/parameter values for the DES application. We have fitted theoretical distribution to data on historical internal UPC arrivals. This fit is used in the DES model to generate workload reflecting the real system. We selected the period 2018-2019 to represent a baseline workload scenario. For this time period, the Normal distribution is selected. As not every arriving UPC follows the same route through the process, we have defined percentages that are used in the DES model to determine which path a UPC will follow. We also inspected UPC arrivals in case of a new retailer integration based on data of the Bol.com integration. In such a scenario after the initial overflow workload, the mean and standard deviation of the workload are expected to increase. We have also seen that with an increasing workload mean, the percentage of eligible UPCs decreases. To increase realism of the DES model we have analysed the type of Keycats that the arriving UPCs belong to. We have constructed a Cumulative Distribution Function to include in the model, that serves as an input in the DES model to generate jobs with an assigned product category (Keycat) according to the real life distribution.

To model the human resources that process the jobs in the operational process of IRI, internal data on historical processing times is analysed and again distribution functions are fitted to be used in the DES model. We have found that for the tasks performed at IRI, the lognormal distribution has the closest fit (from the set of theoretical distributions available in the Java Simulation Library). Lastly, we have substantiated our assumptions for the theoretical distributions that are used for the activities where no data was available. Based on our knowledge of the process, advice from process experts and the fact that many activities in the Coding and DBA department bear close resemblance.

7 - DES MODEL DEVELOPMENT

This chapter will present the steps and methods taken for the development of the DES model. It will provide argumentation for the selected general purpose programming language, as well as the simulation library used. Other elements of the chapter include the conceptual model, the data used (from chapter 6), notes on implementation, integrated process alternatives and expected model limitations.

7.1 - Programming language selection

In the literature review of chapter 4, a case is made for the use of the Java programming language for the construction of a DES model. Java is a programming language that was first introduced in 1995, and has still a strong user base despite its age, mainly because many applications in big companies continue to rely on it. With the construction of a completely new application, a case can be made for the use of a more modern language. Such a language is expected to be better suited for a modern programming environment, being able to better leverage modern technology and be better future proof. As becomes clear from subsection 4.3.2, Java is very well suited for DES applications. However, with the additional need for a more modern language, the Kotlin language is the better choice. Kotlin (first introduced in 2016) is a language based on the Java Virtual Machine, which means it is essentially a modern version of Java. It contains all functionality of Java, including library support and compilation to Java bytecode. Advantages include more concise code and better readability, which helps with development time and credibility in the organisation. Another advantage of Kotlin is its seamless interoperability between all different systems including mobile, web and pc, with native UX. This could be useful in extending the DES application to more platforms in the future to increase usability and practical value. Just like Python, Kotlin is a combination of object oriented as well as functional programming, in contrast to Java which is solely object oriented. An added advantage is that it is now promoted as the primary language for Android development by Google. A disadvantage of Kotlin is that it takes about 15% longer to compile, and being a newer language there exists less support. These two disadvantages are relatively minor, since when an application is written, compilation has to be done only once and Java code can be automatically converted to Kotlin code and the other way around.
For the development of the DES model in this research we will use the Kotlin programming language, both because of the advantages over Java mentioned above, but also because the researcher has more experience in this language.

7.2 - Open-source DES library

To avoid unnecessary work and reinventing the wheel, in almost all programming projects it is advisable to make use of available (open-source) libraries. For this research the Java Simulation Library (JSL) was chosen, which was developed by professor M. Rossetti and was first released in 2005. This library provides an extensive range of general classes and interfaces for random number generation, statistical collection, basic reporting, and discrete-event simulation modelling elements such as stations, resources, entities, queues and the event-list. We have chosen to use this library over other libraries because of the following facts:

- While it was first released in 2005, the latest version stems out of 2022, so it is very up to date.
- There exists excellent and extensive support in the form of a complete book that explains the principles and basics of the library: Simulation Modelling Using the Java Simulation Library, which is used as reference material for the development of the model for this research. Also a very in depth github page is available, as well as a set with example models.
- The library utilises the event view approach to discrete event simulation, which corresponds to the most intuitive way to model the IRI process.

7.3 - Conceptual Model

The simulation model of the operational system within IRI is modelled with the use of the Java Simulation Library (JSL) and the accompanying book of Manuel D. Rossetti implemented in the Kotlin programming language. The JSL uses the event-view approach to discrete event simulation. The terms and concepts that are used with this approach will be conceptualised below with respect to the system within IRI to be modelled, the definitions used are all as defined in the book Simulation Modelling Using the Java Simulation Library of M. D. Rossetti (2022 -latest revision).

7.3.1 - System

A system is a set of interrelated components that act together over time to achieve common objectives. The system that will be simulated for this research are the processes of and the interaction between the Coding and DBA departments with regard to the processing of unclassified EANs in one IRI production period of 4 workweeks. The objective of this system is to process as much as possible of the arrival amount of EANs, avoiding high workload variation (between weeks and employees). For the system, the queueing representation distilled from the complete BPM model of IRI (slightly simplified for modelling and relevancy purposes) of both the Coding department as the DBA department from chapter 5 is used.

7.3.2 - Parameters

Parameters are properties of the system in the form of quantities that do not change. These are typically quantities (variables) that are part of the environment that the modeller feels cannot be controlled or changed. Parameters are typically model inputs in the form of variables. Parameters that will be used to model the system presented above are number of EANs (jobs) entering the system (as lognormal distributed variable), the distribution of these EANs over product categories, the number of attributes (to be coded) per category, processing time of employees (resources), the number of employees per station (full and in training), certain percentages (only Keycat job, feedback received, auto-engine Keycat, ean's with important and proactive label, hit rate), the days on which certain activities occur (ean arrivals, Keycat days, Coding days, Placement days), workload division rules and rules when a jobs is considered late.

7.3.3 - Variables

Variables are changing quantities that are properties of the system (as a whole) that change or are determined by the relationships between the components of the system as it evolves through time. In the system presented above the variables consist of:

- the jobs processed
- the cycle time
- the number of jobs that are late
- jobs in queues and in service
- the number of jobs present in the system (EANs and STUBs)
- number of rework events
- categories present in the system
- number of jobs awaiting feedback.

7.3.4 - System State

The system state is a "snapshot" of the system at a particular point in time characterised by the values of the variables that are necessary for determining the future evolution of the system from the present time. The minimum set of variables that are necessary to describe the future evolution of the system is called the system's state variables. The state

variables of the system consist of the number of jobs present in each queue per resource and being processed per resource at each moment in time, this includes the number of resources present per station per department.

7.3.5 - Entities

An entity is an object of interest in the system whose movement or operation within the system may cause the occurrence of events. In this case the entities are the individual jobs that must be processed per station per department which are the EANs entering the system and the STUB files that consist of attached newly processed EANs of a certain Keycat (category).

7.3.6 - Attributes

An attribute is a property or variable that is associated with an entity. The attributes associated with the entities of the system are the product category of the EAN, the priority, the completion state (new, Keycatted, coded, placed, NIL created, submitted), the rework status (on-track, late), rework event status (if the jobs is an internal rework request), the number of attributes and the number of days a station hasn't been able to finish the job.

7.3.7 - Events

An event is an instantaneous occurrence or action that changes the state of the system at a particular point in time. The events of the system are the arrivals of a new batch of EANs in a production period, which flow directly to the individual resource queues via a workload sharing rule, the arrival of feedback after a NIL report has been sent, and the arrival of an internal rework request. Other events include when the following activities are started and finished: Keycatting, Coding (which launches the GUTU compare function, that attaches EANs of the same category to a newly made STUB (file), placing, NIL reporting and submitting. An activity is defined as an interval of time bounded by two events (start event and end event).

7.3.8 - Resources

A resource is a limited quantity of items that are used (e.g. seized and released) by entities as they proceed through the system. A resource has a capacity that governs the total quantity of items that may be available. If an entity attempts to seize a resource that does not have any units available it must wait in a queue. Resources in the system are the employees per station, the stations of the system include the Keycatting station and the Coding station (which are managed by the same employees in the base model, so the same resources), the Placement station (managed by the Placement team) and the NIL reporting and submitting stations (managed by the Placement support team), also variants on these employees can be assigned in the form as employees in training.

7.3.9 - Queues

Queues are locations that hold entities when their movement is constrained within the system. The current system in place at IRI is modelled in a way that each resource is a single employee and as such has its own job queue, there are some intermediate queues constructed in the simulation application to allow for jobs to wait on a new activity to start on a specific day.

7.3.10 - Future Event List

The future event list (or event list in short) is a list that contains the time ordered sequence of events for the simulation. This event-list is handled by the JSL under the hood.

7.4 - Additional DES model characteristics

Besides conceptualising the process of IRI within simulation concepts used within the JSL, we also need to define additional characteristics.

7.4.1 - Simulation horizon & tick precision

To make sure the simulation model is not more difficult than necessary, a finite simulation horizon of 1 IRI production period of 4 weeks is chosen. Rossetti (2008) states that finite simulation models/systems present less complications in a simulation study then continuous models including the definition of steady state behaviour or or the determination of the warm-up period. Within this simulation horizon, the productive work hours are simulated with a precision of 1 second. This tick precision is used to account for some jobs that are measured in seconds, rather than minutes. Because of the event-based nature of the simulation, not every second has to be simulated but only the set of events that can occur with a precision of 1 second intervals. The hours in a production period of 4 weeks that no work is performed such as evenings and weekends is disregarded and productive work hours are joined together in one single run to avoid unnecessarily complicating the model.

7.4.2 - Determination of the number of replications for the DES model

The DES model is dependent on random number generation by random number generators. Because of this it is necessary to determine the number of times the model needs to run (number of replications) to provide statistically relevant results. The final output of the model will be the average of the output of each replication. The number of replications can be determined in the following ways:

- Checking a statistical accuracy terminating criterion after each replication
- Using a predetermined replication value determined with the use of an approximation method

The JSL has the ability to automatically determine the required number of replications for a response variable in the model and a maximum deviation of the (across replication) sample mean. This, however, is done by checking if the preset required statistical accuracy is reached after every replication. The simulation is terminated and the required number of replications is displayed if the accuracy is reached, another replication is started if it is not. This technique is more exact than a replication number approximation technique, as no fixed predetermined number of replications is used but the number of needed replications is determined on the specific simulation/experiment. However, the implementation of this technique might jeopardise the practical value outside of the reach of this research. The required number of replications rises exponentially with the required statistical accuracy approaching 100% (the closer the required accuracy is to 100% the tighter the bound). This means that if this method is implemented in the model and required bound can be inputted manually to calculate the required number of replications, there is a high risk of a too tight bound and an extremely long running simulation without usable results (too many replications to use in practice). These risks can be mitigated by limiting the running time of the simulation, however the fact that relevant (statistical) knowledge is necessary to properly make use of such a function we opt to not implement this function in the model at this time and give a general recommendation for the best number of replications using the normal approximation of the half-width ratio method to determine the number of replications.
To determine the desired number of replications based on the normal approximation of the half-width ratio method an initial pilot run of the simulation has to be done with an arbitrary amount of replications, we choose to run the simulation with a number of 15 replications.

The next step is to determine the estimate of the initial sample standard deviation. M. Rossetti (2008) gives a formula to calculate this value s_0:

$$s_0 = \frac{h_0\sqrt{n_0}}{t_{\alpha/2, n_0-1}}$$

Where h_0 is the initial value for the half-width from the pilot run of n_0 replications. $t_{\alpha/2, n_0-1}$ is the corresponding t distribution value for parameters α and n_0.

For the pilot run with $n_0 = 15$ we get a 95% confidence interval half-width of $h_0 = 277.5323$ when using the Completed EAN Jobs as across replication response variable. $t_{0.025/2, 14}$ corresponds to the value of the t-distribution with a probability of 0.025 and 14 degrees of freedom, which corresponds to 2.14453. This means s_0 is equal to 484.22.

We use $z_{0.025} = 1.96$ to determine the minimum required replications for a range of different margin of error (E) bounds on the precision of the system time of the simulation using the formula $n \geq \left(\frac{z_{\alpha/2} * s}{E}\right)^2$, table 5 shows the required number of replications for a range of E. Figure 24 shows a graph of these values.

E	Required Nr. Replications
70	196.910495
80	150.7595977
90	119.1186945
100	96.48614253
110	79.74061366
120	67.00426564
130	57.09239203
140	49.22762374
150	42.88273001
160	37.68989942

Table 5 - Required nr. of replications per E

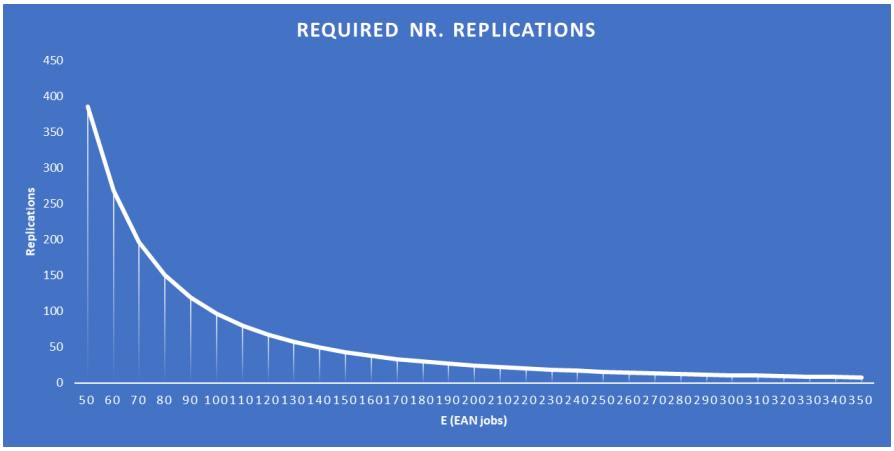


Figure 24 - Graph of required nr. of replications per E

We will set the standard number of replications to 100 to get a good balance between performance and margin of error.

7.6 - Key elements of the Kotlin implementation of the IRI process in the JSL

This section will briefly expand on some of the most important elements of the Kotlin DES application.

7.6.1 - Entities

There are 2 entities within the DES model: EAN and STUB jobs. EAN jobs have a different number of attributes that need to be coded, which is part of the EAN object as constant. Also in the EAN object the location and completion state is registered with the use of variables. Entities can be considered to be "late" which means they have been at the same place in the process for a number of days defined in the late rule. Each time an entity is not completed within a day, the variable "daysNotFinished" is incremented and the priority of the object is increased to make sure these jobs remain at the top of the workload queue. STUB jobs can be seen as containers of EAN jobs from the same category that get created after the Coding job within the process. The event that creates the STUB job is called "GUTU compare" (which

references the real life action). This event occurs at the end of every week (in the weekend). STUB jobs contain a "new adds" list which contains EAN jobs. STUB files often belong to multiple clients, so the actions belonging to the DBA part of the process have to be completed multiple times to complete the STUB job. This "STUB workload factor" is set to the average of 3 (STUB file belongs on average to 3 clients), however it is possible to extend on this as the STUB workload factor can be set for individual STUB categories.

7.6.2 - Implementation of the IRI process

The two most important objects of the JSL that are used in the DES simulation to realistically model the IRI process are EventActions and ReceiveQObjectInterfaces.

An event action (comparable to an event in a general DES) is an instance of the EvenAction object of the JSL and can be planned one or multiple times on the simulation horizon at a certain time. The code within an event action object defines what happens when the event occurs. The arrival of EAN jobs in the model or the receival of client feedback are examples of event actions used in the DES application. Further use of the event action is more abstract and done to increase how much the model resembles the real process: the IRI process implemented in the DES application consists of separate workdays on which separate departments (sometimes with the same employees) perform 1 specific action. As mentioned before the tick time of the simulation is 1 second and the time between two workdays where no work is performed is disregarded. To make sure the right activities are performed on the right days, events are planned at the start (and sometimes the end) of each workday. These events make sure that the workload which was completed in a previous step of the process and held in an intermediate queue gets sent to the queues of the next station in the process. Also, queues of stations get emptied at the end of a day and sent to the right intermediate queue (not connected to a station).

ReceiveQObjectInterfaces can be seen as abstract stations that facilitate the right flow of jobs coming from a previous stage in the process. The dispose (drain) objects of the DES that make sure that completed QObjects are removed from the simulation and simulation statistics are updated are instances of the ReceiveQObjectInterface for example. To make sure a job does not go to the next station but to the right intermediate queue after it is finished by a specific station is also handled by instances of the ReceiveQObjectInterface. This is necessary to make sure the next stage of the process can not start on a day it should not perform the activity. These interfaces are connected to their respective station, and receive QObjects (jobs) after the server of a station has completed it.

7.6.3 - Mistake & Rework modelling

To provide an extra layer of realism, on top of the normal process, a simple form of mistake & rework modelling is integrated. From the data it becomes clear that sometimes (around 5% of jobs) certain completed tasks are done again (such as assigning a new Keycat or changing coded attributes). This number is most likely not high enough to have a significant effect on performance of the system. We do however choose to incorporate this to achieve the closest model resemblance to the real system as possible without having to sacrifice performance (this part just requires drawing a few extra random numbers and rerouting some jobs). The percentages from the data are used in the DES model to register after a task is completed if it is done right the first time or not. If a task is not done right the first time, this will not get noticed immediately, but one or multiple stations later in the process will be checked if a previous job stage has been completed with success. If a job gets identified as not being done correctly it will get sent to a rework queue of the station responsible for the task which is not done right. These rework queues are processed on days specified. On top of the jobs stages which can be completed unsuccessfully, there occur internal and external rework requests as well outside of the normal job flow. These requests are generated separately and also land in the separate rework queues mentioned before.

7.6.4 - Stations

The station object present in the JSL library has been modified for each major stage in the process: Coding (Keycatting + attribute Coding), Placement, Placement Support (Placement validation, NIL reporting, feedback, submission). Modifications include the use of the right QObject natively (EAN or STUB) and the service time calculation. An instance of a station represents a single station at a certain minor stage with a single employee as a resource. Different stations can have the same employees, which is the case at the Coding major stage in the baseline process that has the same Coding employee for both the Keycatting and attribute Coding stations.

7.6.5 - The main class

The main class is the class where the application is run from. In the IRI DES application this class contains code to facilitate a console based user interface, where the user can determine the values of certain process characteristics, the number of experiments and whether to write output to a .csv file.

7.6.6 - Integration of process alternatives

Different process alternatives are integrated in the code of the DES model. This means a different process alternative can be selected just by setting an input parameter, so no code has to be modified and no other applications have to be run. An example of an integrated process alternative is the use of a single workload queue for every station (in the base process each station has its own workload queue) or the separation of the Keycatting and Coding task (using different employees at each step). Boolean parameters can be set to determine which process modification should be made and with the use of if-statements the code for the selected process is run.

7.7 - Conclusion on the Kotlin implementation of the DES in the JSL

The DES application that was constructed for this research was written in Kotlin (see section 7.1). This is not a popular or commonly used language for DES modelling or even for the development of computer applications. The Kotlin programming language is primarily popular for Android development (especially after the support of Google). The language is significantly less used for PC console/GUI applications than Java. What the development of the DES model in this research has shown is that there is no reason the use of Kotlin should be limited to Android development. Kotlin-Java integration has worked without problems, as the Java Simulation Library (written in Java) works seamlessly with a Kotlin based top level application. The limited support on the internet (compared to Java) also is not a problem as Java code converts (almost always) automatically to Kotlin. Another remark that can be made is that a lot of boilerplate, repetitive code could be avoided resulting in less development time, better readability and more compact code. This is one of the advantages of Kotlin over Java.

The implantation of the IRI process has been set up with modifiability in mind. This has become useful with implementing the process alternatives presented in chapter 8, but also enables this DES application as an excellent starting point for further development (such as an even higher degree of resemblance to the real process, inclusion of more data or the integration of more process alternatives/scenarios). This first application could also provide a good starting point for the use of DES in other IRI countries (where the process might deviate). The JSL provides an excellent foundation for modelling the most intricate and unusual processes with high resemblance if some creativity is used. For this research, a barebones console based UI selection menu is included to provide the option to analyse a very large amount of different scenarios and model parameters without having to dive into the code. Although this works well to improve usability of the program, the development of a complete graphical user interface based on JavaFX could be an interesting, relatively easy to implement addition to improve user friendliness and increase the use and applicability outside of this research. Appendix D contains pseudocode for some of the most important and interesting elements of

the JSL DES implementation in Kotlin. The full Kotlin code can be found in the following Github link: https://github.com/egsgheijs/IRI_Mercury

7.8 - Chapter conclusion

In this chapter we have provided the steps we have taken to implement the process of the Coding and DBA departments in a DES computer application. We have selected Kotlin (a modern programming language similar to Java) as the programming language to use. This language has a number of advantages over Java (which is often used for DES, see subsection 4.3.2) such as better readability, less development time (less boilerplate code), easier GUI creation, improved platform interoperability (desktop, mobile, web) and better expected future proofness (promoted by Google).

To avoid further unnecessary development time, the Java Simulation Library (JSL) is used as the open-source DES library to build the DES application with. The JSL is complete, up to date and uses the event-view DES approach. The event-view is often used to model production systems and it is the most intuitive way to model the process of IRI.

Based on the JSL, a conceptual model is defined to serve as an abstract representation of the processes to be models and which is used in the programming of the DES model. The DES model simulates a single IRI production period of 4 weeks with a precision of 1 second, where only workdays of 8 hours are considered. After construction of the DES application, a set number of 100 replications is determined to provide results with an acceptable maximum deviation and running time. A set number is used to reduce the usage learning curve for when the application is used in practice. For the interested reader, Section 7.6 and 7.7 expand on some more in-depth implementation details.

8 - PROCESS ALTERNATIVES ANALYSIS

This chapter presents the process alternatives integrated in the DES model and their performance against the baseline situation. The analysis of the process alternatives will be done based on a single effect analysis. To determine this effect, the percentage difference of the performance indicators (defined in Section 3.1) compared to their value in the baseline scenario is used. The varying parameter in a process alternative will be set at a few different levels deviating from the baseline value. Each subsection will present the results of the comparison of the process alternative with the baseline scenario. The Excel file containing the full simulation results and accompanying graphs can be found in the aforementioned (page 54) Github page. A subsection will first introduce the process alternative, then discuss some of the most interesting results and conclude with one or more key insights that can be drawn from the results. These key insights are subjective and the results could be interpreted differently based on what performance indicator is deemed more important. The following effects will be analysed in the process alternative analysis of this chapter:

- Impact of full employee numbers at a certain stage of the process
- Impact of employees in training at a certain stage of the process
- Impact of employee specific workload queues vs one general workload queue
- Impact of different workload division rules
- Impact of UPC arrival variation level
- Impact of different workload arrival moments
- Impact of changing days when tasks are performed
- Impact of AutoEngine Keycat percentages
- Impact of performing Keycat & Coding task together instead of separately
- Impact of different employees for Keycat and Coding tasks
- Process requirements of a new retailer integration in scenario based on Bol.com data

These process alternatives will be compared to the DES model output of the baseline scenario. The output of this analysis is an indication of whether a process alternative is expected to have a positive or negative effect on the performance indicators. The values of the input variables of the baseline situation are based on the stable production scenario of the period 2018-2019 and have been obtained from the data in chapter 6. The specific values used for the baseline process are presented in Appendix.

8.2 - Impact of full employee numbers

In this section we analyse the effect that modifying the amount of employees at different stations has on the performance of the total system. Per station the number of employees will be set at different levels and the percentual change of each performance indicator compared to the baseline is shown in the corresponding bar graphs. Sometimes performance indicators showing less than 1% change will be left out if they are not deemed important for understanding a process alternative..

8.2.1 - Coding employees

In this process alternative we analyse the effect of Coding employees based on three levels: 17 (four more than baseline), 9 (4 less than baseline) and 5 (8 less than baseline). We can see in Figure 26 that while the total system time of jobs does not get affected significantly, the utilisation of the Coding employees (who are responsible for both the Keycatting and Coding task) does get affected a lot. Higher utilisation of employees means that these employees have a higher workload per employee. It will have to be decided by management if higher or lower workload is desired. Interestingly, adding 4 more Coding employees reduces the utilisation by a lower percentage than removing 4 employees increases the

utilisation. This indicates a non-linear relationship between the utilisation of Coding employees and the number present in the system. Going from 9 to 5 employees increases the percentage difference to the baseline performance from +40% to about +160%. The extreme (expected) negative result that the reduction of the number of Coding employees by 8 has on the system, also becomes clear from Figure 26 that displays the change in number of jobs in each workload queue and stage of the process. We do notice that the decrease in Coding employees has a significantly stronger effect on the number of Keycatting jobs than the number of jobs present at the Coding stage. One reason for this is the fact that on average only 40% of jobs get to this stage. Also the average number of EANs present at the Coding stage is generally lower (around 5 jobs) than the Coding stage (around 75 jobs), so per job change a higher percentage change is expected at the Keycat stage.

We can conclude from these results that by modifying the number of Coding employees, it is expected that both system time of jobs as the number of EANs that are submitted (jobs that go through the whole process) are not really impacted. However, the workload of Coding employees is strongly influenced, where reduction of employees has a stronger effect than adding employees. We therefore think that with this workload, modifying the number of Coding employees is not necessary.

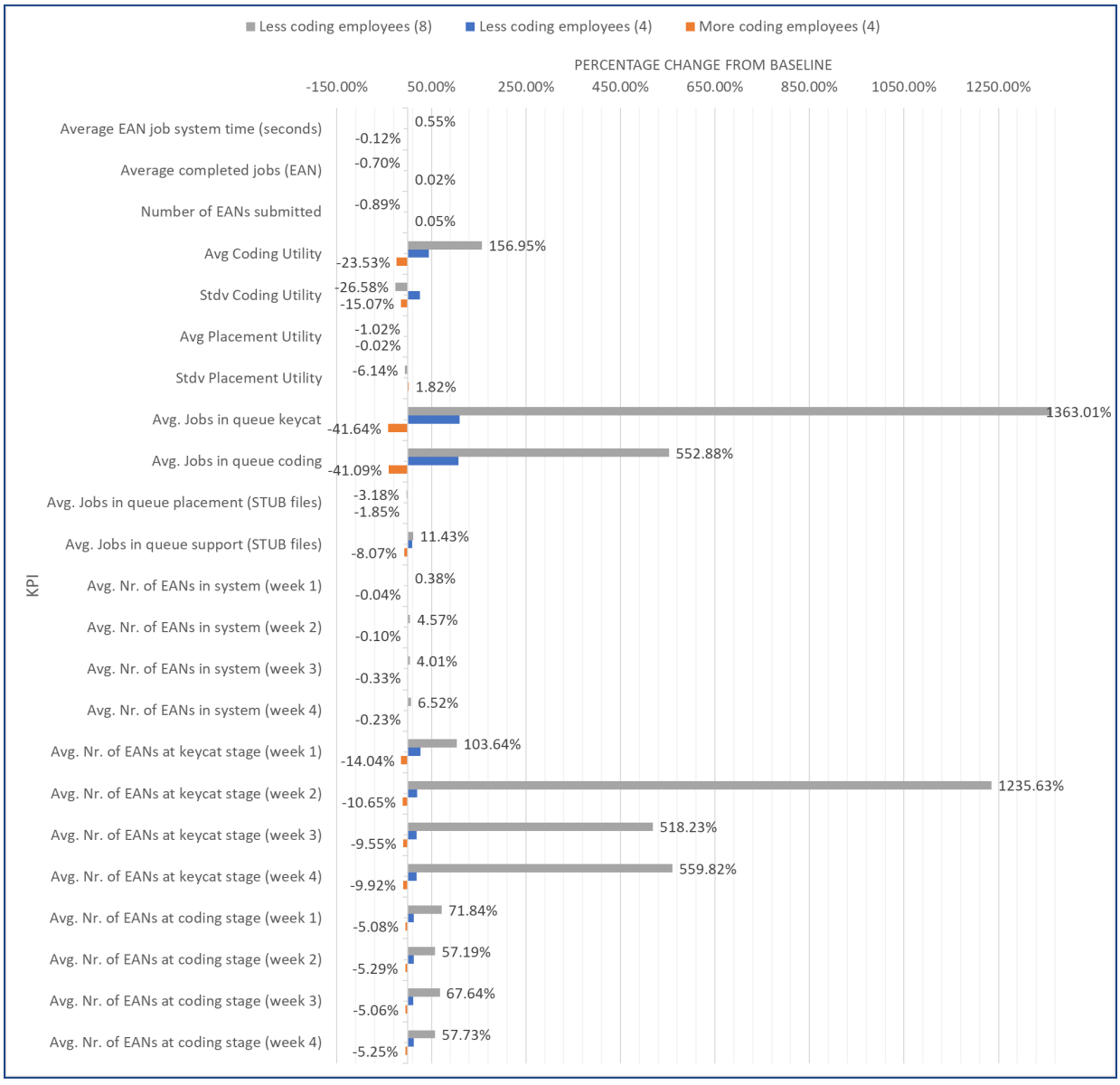


Figure 26 - Percentage change of varying Coding employees compared to baseline

8.2.2 - Placement employees

In this process alternative we modify the number of employees at the Placement stage in steps of two (as the baseline consists of 5 Placement employees). In Figure 27 we see the significant effect of modifying these employee numbers on the total job system time, where adding 2 extra has a stronger positive effect than another 2. Removing 2 employees is not recommended as the negative effect of this is higher than the positive effect adding 4 employees has. Even though system time is reduced with adding Placement employees, no real significant change is expected in the number of EANs that can be submitted. We can also see that the addition of 2 Placement employees has about the same effect on percentual change of utility as the addition of 4 Coding employees. Also note the effect modifying the number of Placement employees has on the standard deviation of the standard deviation of employee utility of the next stage (Placement support). We can see a similar effect on the number of jobs present in the station as the scenario of 8.3.1, although we do note the quite significant increase in jobs at the Placement support stage when increasing the number of Placement employees. Less percentage change is expected in the number of EANs at the Placement stage compared to the percentual changes we saw in the scenario of 8.3.1, again partly because of the discrepancy between the jobs present in this stage (around 1200).

We can conclude for this scenario that Placement employees are more important in the system if we look at the total job system time, but that changing this parameter alone cannot account for an increase in job submissions. We can also identify comparable results to utility and number of jobs in the queue as with varying Coding employees, although 2 Placement employees account for about the same effect as 4 Coding employees. A change in this department has significantly stronger effects on the next department as with changes in the Coding department.

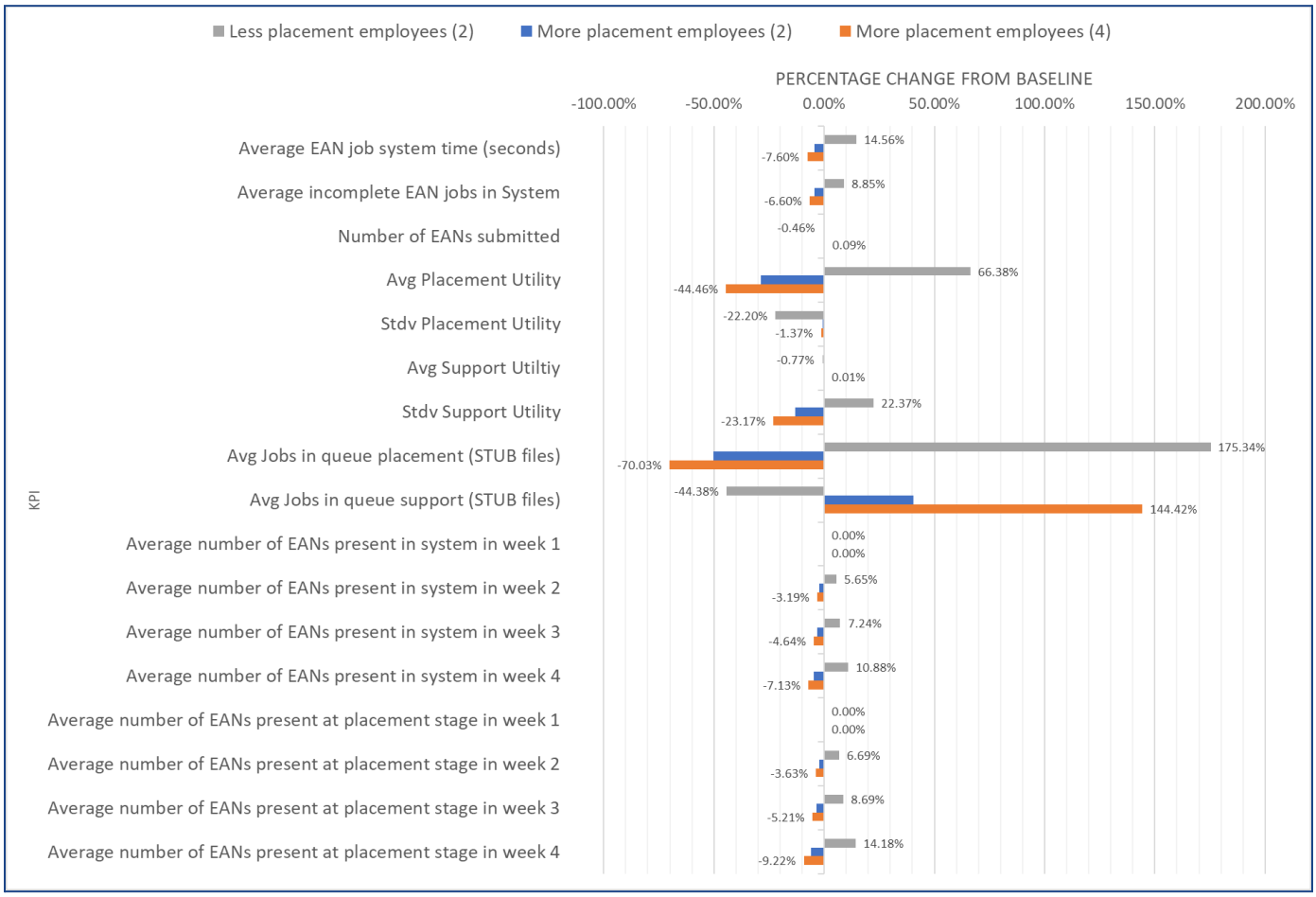


Figure 27 - Percentage change of varying Placement employees compared to baseline

8.2.3 - Placement Support employees

In this process alternative we modify the number of Placement Support employees. Modifying the number of Placement Support employees has the least amount of percentage change on the total system compared to modifying the other types of employees as can be seen in Figure 28. As the utility of Placement support employees in the baseline scenario is considerably lower than the utility of both Coding and Placement employees we do not consider more than 3 (baseline number) Placement support employees. The main effects of reducing the number of Placement support employees (shown in Figure 28a and 28b) is again similar to the scenarios seen in 8.3.1 and 8.3.2, with the expected (non-linear) increase in utility and increase in average number of jobs in the queue. We see a similar percentage increase in the number of EANs present in the Placement support stage of the process as we have seen at the Placement stage in the scenario of 8.3.2. The extreme difference between percentage increase between jobs in the queue and number of EANs in the corresponding stage (Placement support) can be explained by the fact that the number of jobs in the queue is counted not in EANs but in STUB files containing a number of EANs. Nevertheless, while the percentage change is extremely high (+900% to +18800%), the actual number of jobs in the queue is still relatively low compared to other departments. This also holds for the utility of the Placement Support employees compared to other employees.

We can conclude that it might be interesting to reduce the number of Placement support employees performing these tasks related to the submission of EANs. Placement support employees have a great variety of different tasks, where in the simulation model we have excluded those activities not directly related to the submission of EANs. This means that if tasks were to be divided, with for example one Placement support employee responsible for the tasks related to EAN submission, other support employees could perform a set of the other tasks and some efficiency gains could be gained.

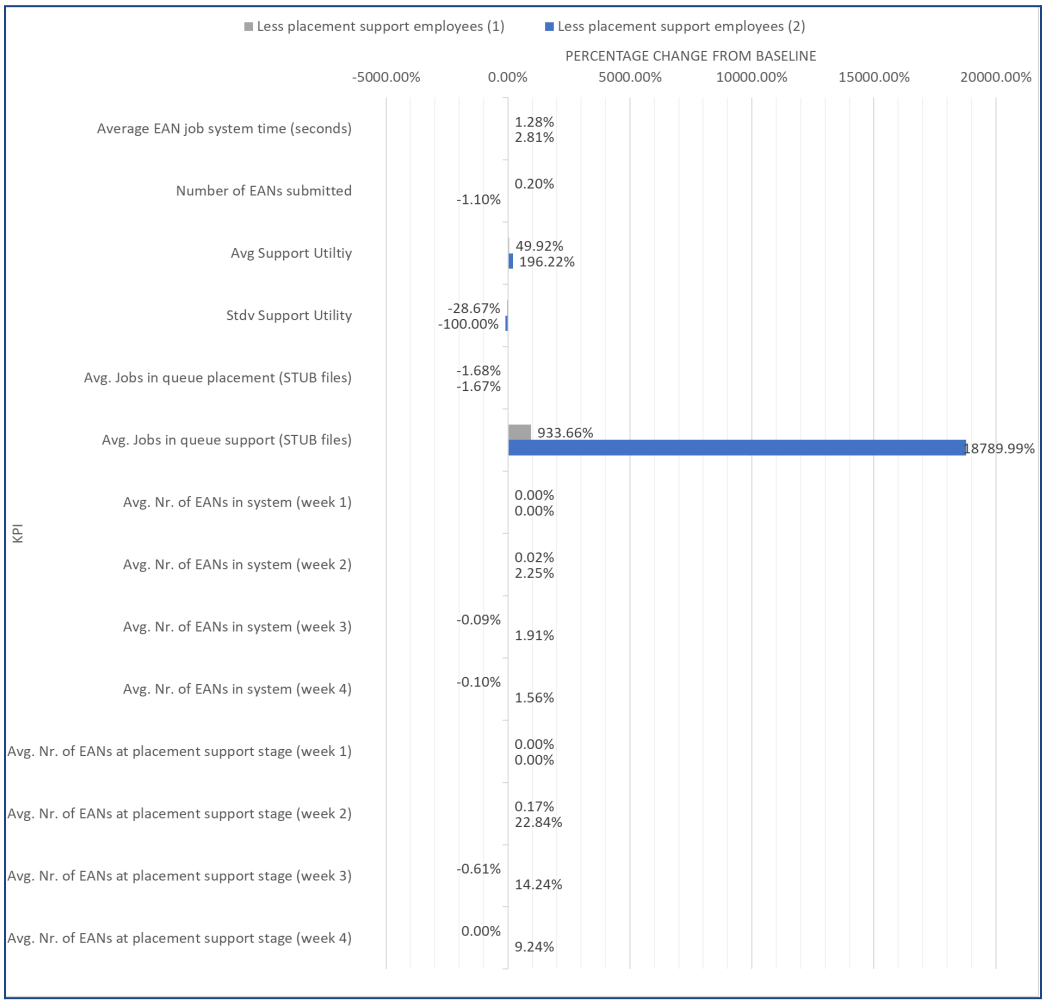


Figure 28 - Percentage change of varying Placement support employees compared to baseline

8.3 - Impact of employees in training

In this section we will analyse the effect that employees in training have on the performance of the system. Employees in training are defined as employees that have newly joined the company and are learning their specific job by doing. These employees have an adjusted distribution for their service times as defined in 6.5.3. The analysis will be done in two ways:

- By substituting a part of the full employees of the baseline numbers with a number of training employees. This way we will get a good understanding of the expected performance loss.
- By adding a number of training employees to the normal employee base of the baseline scenario. This way we will get an idea of the expected increase in performance of the system when in a short period of time new employees have to be attracted.

8.3.1 - Substitution with employees in training

If we look at overall system performance, mainly to total job system time and number of EANs submitted, we note that no real impact is seen by including training employees in the Coding department, but total performance does get significantly impacted by including employees in training in the Placement department. Also, including employees in training in both departments, we see an decrease in the total process performance (higher job system time, less EANs submitted). We assume that this performance hit is, again, mainly coming from the employee in training at the Placement department. Including employees in training in both departments does therefore not introduce an additional performance hit.

When looking at employee utility in Figure 29 on the next page, we see that including 2 employees in training in the Coding department slightly increases normal employee utility but at the same time reduces the standard deviation of the utilisation per employee. An additional 2 employees in training even more than doubles the percentage reduction in the standard deviation of the utility of normal Coding employees. The performance in utility of employees in training is based on the percentage difference of their utilisation with the utilisation of normal employees of the baseline scenario. We see that, even while the performance of employees in training is set at 60% of a full employee, in practice their utility is only about 120% more than a full employee. Also within the Placement department the addition of employees in training is reducing the expected standard deviation of the utility of normal employees at a slight cost of a higher average.

If we look at the number of jobs in the system we see that overall only adding employees in training in the Placement department have an effect on the number of jobs in the total system per week. This action has the effect of increasing the amount of jobs at the Placement stage with an increasing trend over the 4 production weeks, while the Placement support stage experiences less overall jobs (gradually decreasing over the 4 weeks) which can be accounted to jobs generally staying longer at the Placement stage. Adding employees in training at the Coding stage effectuates with a peak at the second week at the Keycatting stage and an increase that remains constant over the four weeks at the Coding stage.

We conclude that the substitution of normal employees with employees in training in a department has only a slight effect on total system performance with the greatest hit from substituting Placement employees. Also, it must be noted that normal employees are expected to experience a slightly higher workload. The most important insight we get out of the analysis of these results, is that substituting full employees with training employees in the Coding team is less disadvantageous for the total process performance than including them in the Placement team.

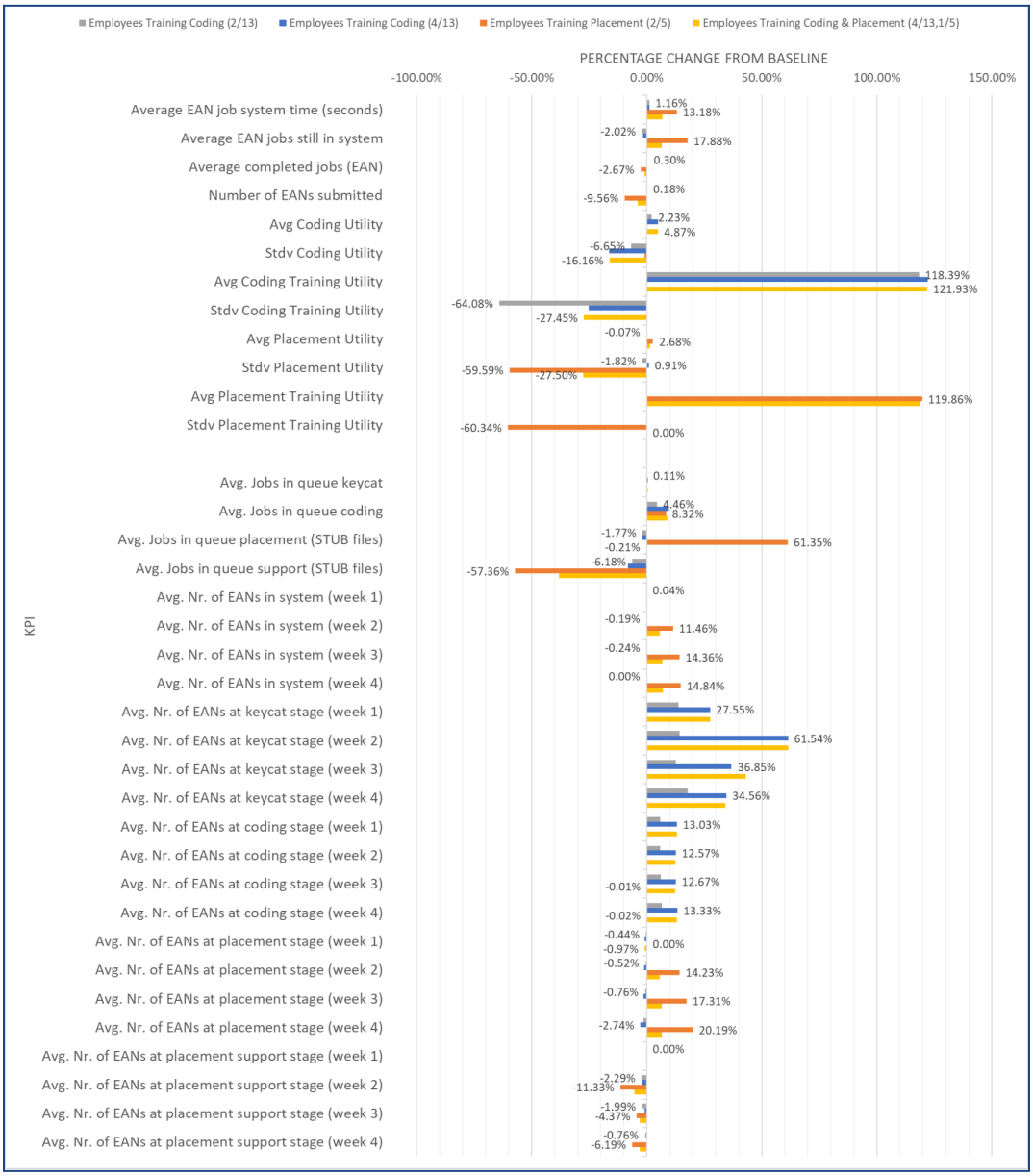


Figure 29 - Percentage change of varying the number of employees in training substituted compared to baseline

8.3.2 - Addition of employees in training to the base number of normal employees

While subsection 8.3.1 showed the performance hit when substituting normal employees for employees in training, this subsection will explore the performance that can be gained with the addition of employees in training to the baseline scenario team of normal employees. It can often happen that sudden additional workload is expected, and the existing team is not enough to handle the workload. In such a case, new employees have to be attracted. These employees have not been trained, and therefore are less efficient than a normal employee.

In Figure 30 we see that with the addition of employees in training at the Coding department, the workload of normal Coding employees is expected to decrease, while the workload of Coding employees in training is quite a bit higher when compared to the workload of normal Coding employees in the baseline scenario. This can however be mitigated by assigning less workload to the employees in training. We also notice a decrease in workload standard deviation which is strongest with an addition of 4 employees in training. Figure 30b shows that in general with the addition of employees in

training, a decrease in jobs in the queue can be expected. Extra employees in training at the Placement station will contribute to a slight increase in jobs in the Coding queue. This also becomes clear when we look at the amount of EANs present at each stage per week, where every training employee addition has the strongest effect on the number of jobs present at the Coding stage.

The key insight from this process alternative is that the addition of employees in training at both the Coding and the Placement stage is not expected to improve process performance. The utility gains for the full employees shown here are reduced with assigning less workload to the new employees. Also, the full employees need to spend time on training the new employees.

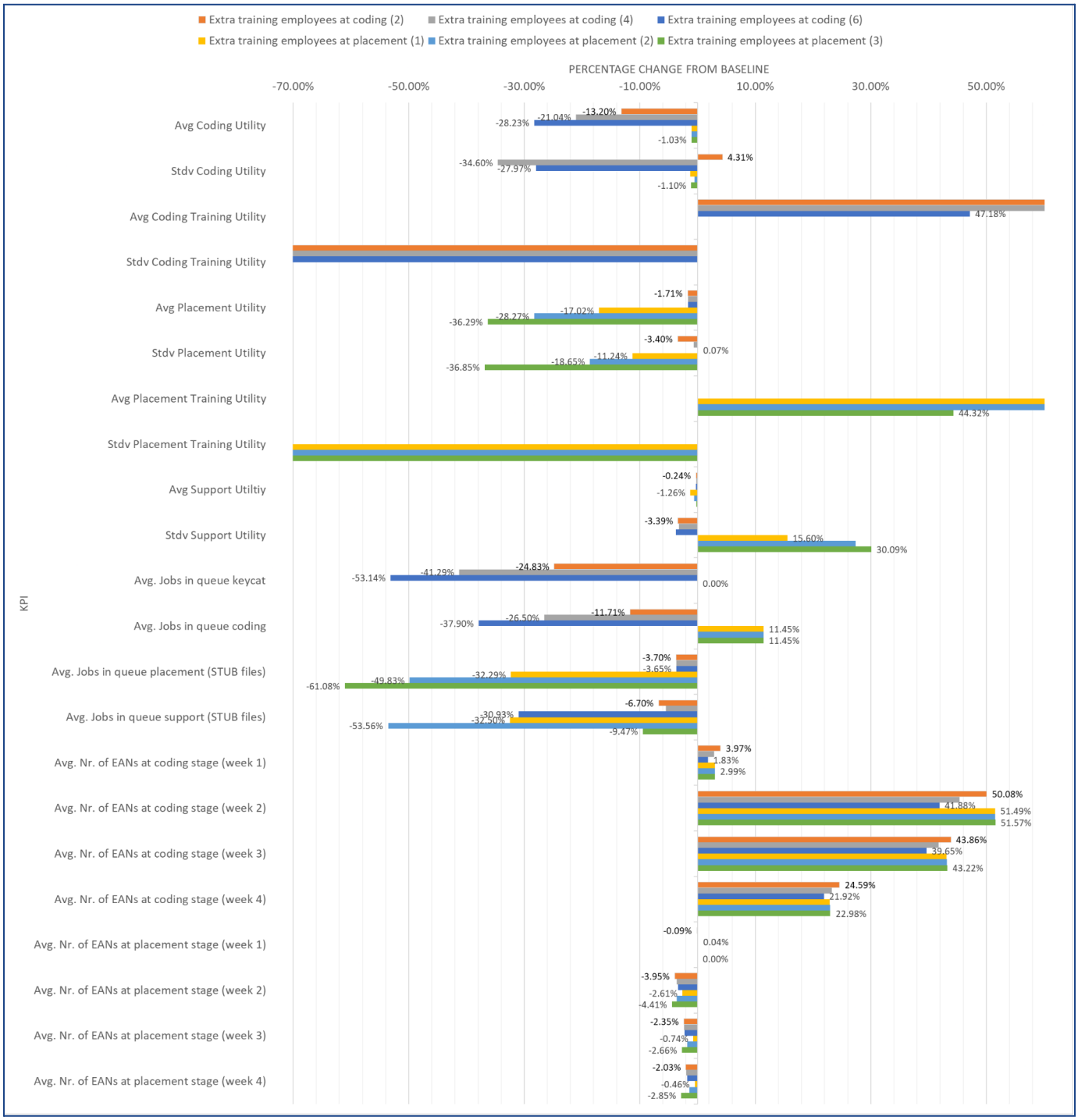


Figure 30 - Percentage change of varying the number of employees in training added compared to baseline

8.4 - Impact of one shared workload queues at stations

In this section we will analyse the expected effects of using one shared workload queue for all employees at a station instead of separate workload queues for each employee at a station. Workload arrives via a workload division rule at the start of a workday. From Figure 31 on the next page we notice the main effect this process alternative has is the significant decrease in utility standard deviation. This is of course an obvious effect as every employee gets a new job from a shared queue instead of a pre-divided set of jobs where a discrepancy between workload is more likely. We also notice the slight decrease in total utility of the station where the change is applied. Despite this, no real change is expected in total system performance as system time and total submitted EAN remains about the same. The main effect we can see in Figure 31b is strong increase in average and standard deviation of jobs in the Placement support queue, as well as the quite comparable increase in number of jobs in the Coding stage for every type of single queue implementation. The -100% of standard deviation of workload in the queue where the change is implemented can of course be explained by the fact that only one queue exists.

In conclusion we can say that this modification is quite successful at reducing workload variation between different employees at a station. This can, in turn, improve working conditions and employee satisfaction without sacrificing system performance. The total system performance gains, which would seem likely, are not likely.

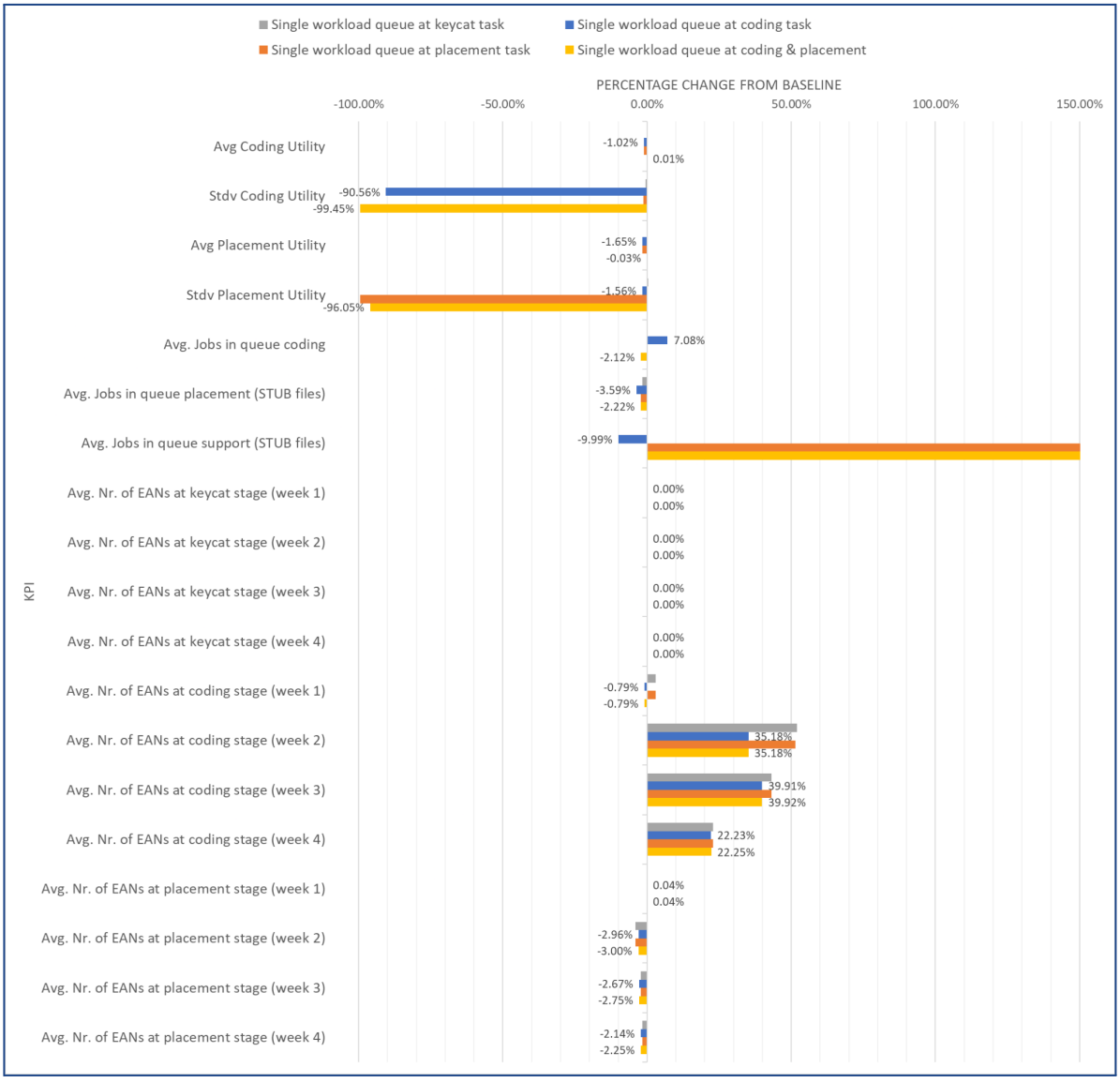


Figure 31 - Percentage change of using shared workload queue

8.5 - Impact of separating Keycat and Coding task

In this section we will analyse the effect of separating the Keycat and Coding tasks into separate teams. In the baseline process, the same group of (Coding) employees performs both the Keycatting as the Coding task. Each task is done on a separate day, and on a day only a single task is performed. In this scenario we introduce a separate group of Keycat employees that will perform only the Keycatting task. The same number of employees is kept as in the baseline, only now a portion will only perform one or the other activity. An advantage of this change is that now on every day of the week both Keycatting as Coding tasks can be performed by the separate teams. Also performing only a single task has the advantage that it is learned quicker and efficiëncy gains in processing time are expected. A downside is that performing only a single repetitive task day after day does not contribute to employee satisfaction.

In this scenario total system performance remains quite stable as seen in Figure 32a, although a slight increase in the number of submitted EANs is expected. Most notably we can see that if only one employee is switched, its workload compared to what it would have been as a Coding employee in the baseline scenario is only expected to increase with about 13%. With switching more Coding employees to Keycatting employees, this workload compared to being a Coding employee is expected to decrease significantly. We do notice that with the introduction of a separate Keycatting team, the workload of the Coding team is expected to increase, while the standard deviation most notably with a Keycatting team of 2 employees is expected to decrease. Figure 32b shows that the expected percentage increase in jobs in the queue dramatically affects the Keycat station, but that the rest of the system only experiences a slight rise of jobs in the queue. In Figure 32c it becomes clear that the expected average number of jobs over the whole system per week are not really influenced, but that the expected number of jobs at the Coding stage over the 4 weeks is expected to rise significantly.

In conclusion, a separation of the Keycatting and Coding task could be quite advantageous. We also think that this process alternative might work even better when increasing the number of data arrival moments and increasing the AUTO ENGINE percentage, this will flatten out the workload and remove some of the workload strain from a possible small Keycatting team. Indeed, with increasing the number of data arrival moments, increasing the AUTO ENGINE percentage, having a team of 2 Keycatting employees and moving Placement support employees to the Placement team, we get an expected decrease of the total job system time of almost 20%, while increasing the number of EANs submitted, and decreasing the standard deviation of the workload of the Coding team with an slight decrease in utilisation if we average the utilisation of both the Keycatting and Coding team.

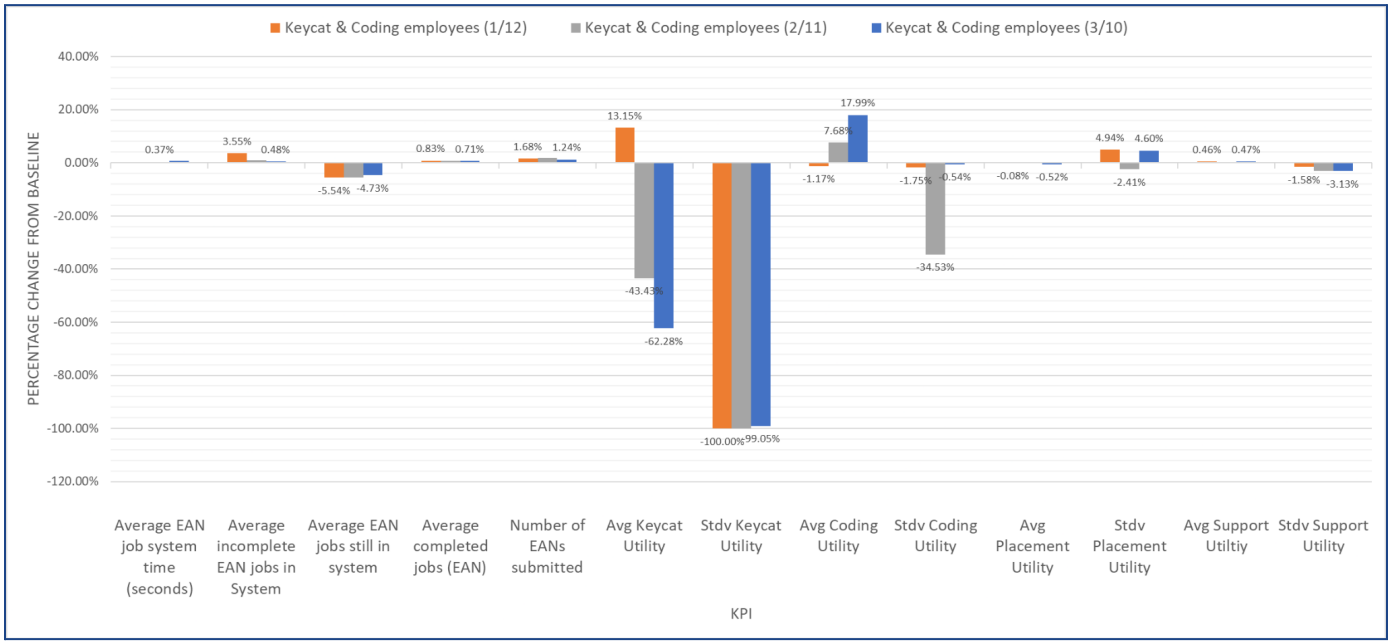


Figure 32a - Percentage change of separating Keycatting and Coding tasks

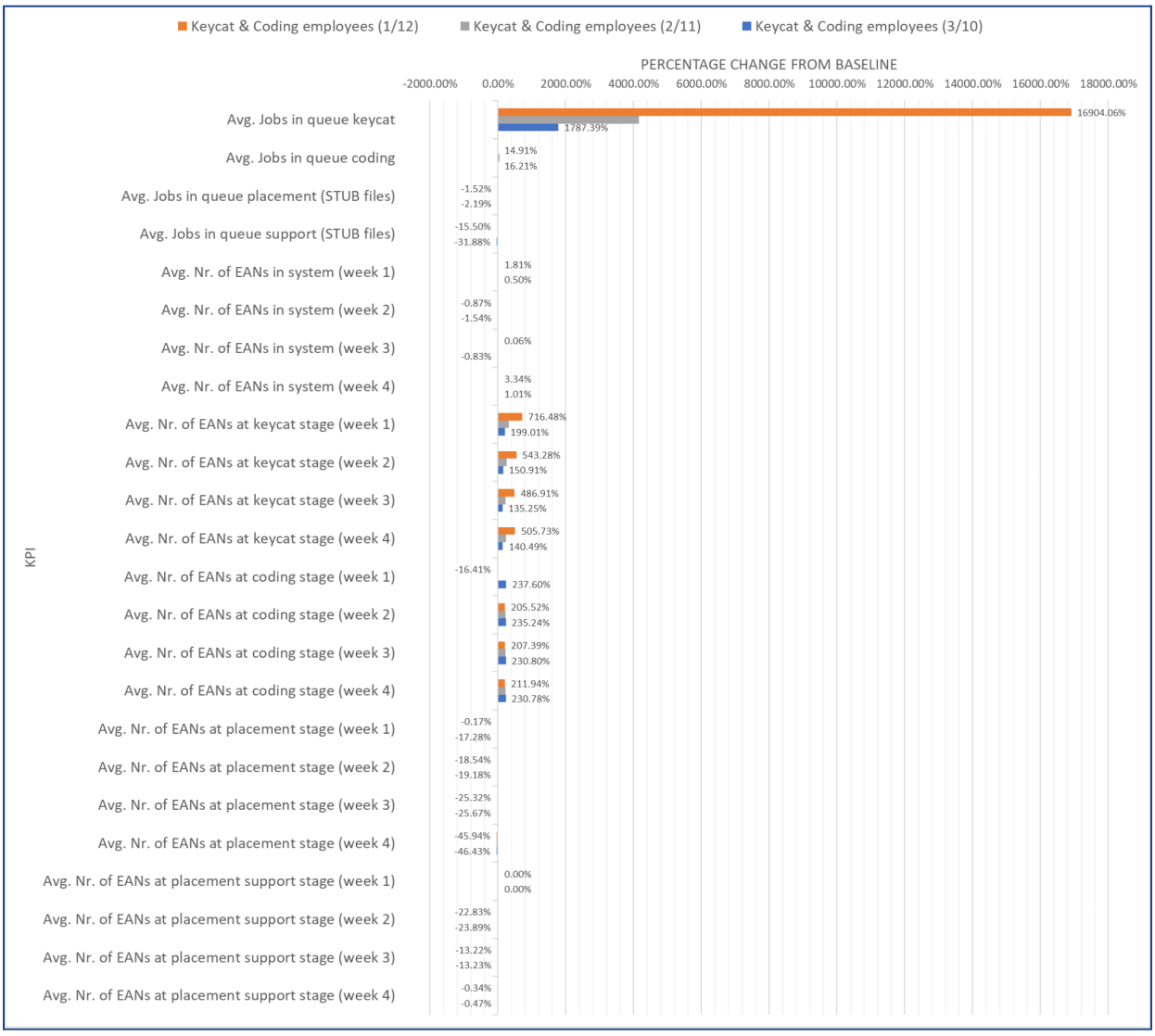
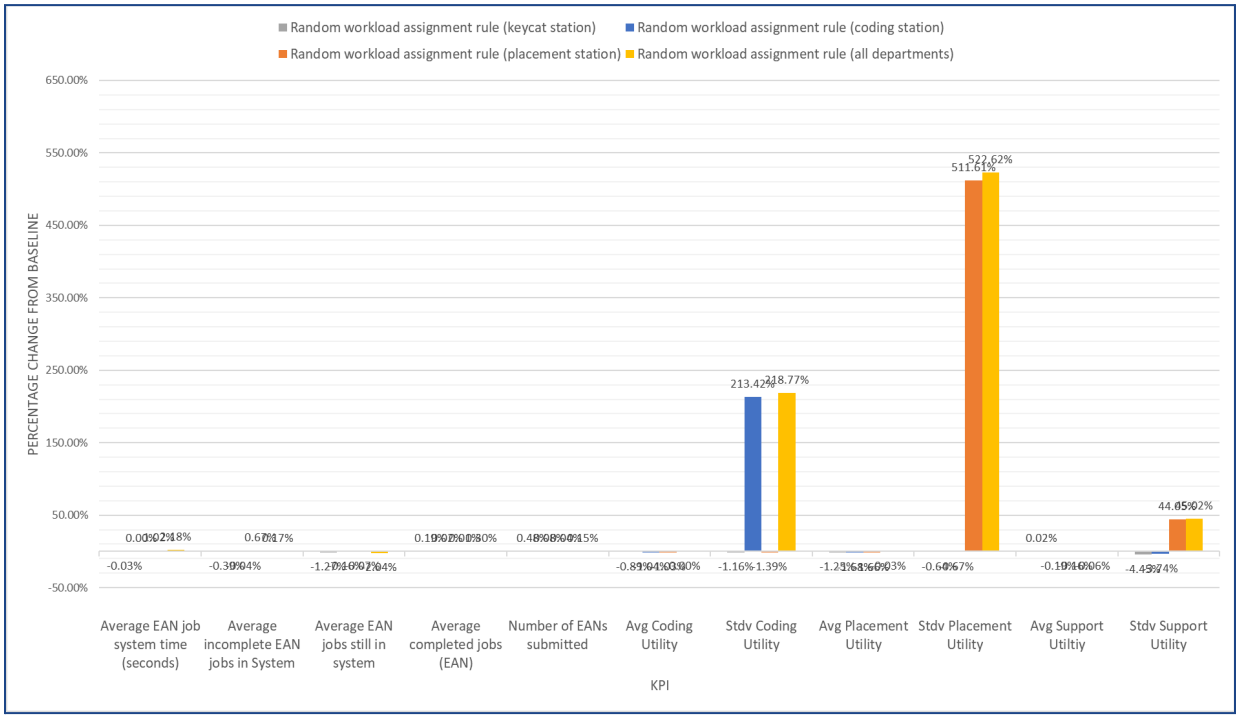


Figure 32b - Percentage change of separating Keycatting and Coding tasks

8.6 - Impact changing workload assignment rule

In this section we will explore the effect the way workload is assigned to each employee affects the performance of the system. For this analysis we use a simple random assignment rule. The baseline scenario divides the workload in a more equal way where the Keycat that occurs the most in the data is assigned to the first employee, the next to the second and so on. We will note that in the real life system the workload assignment might be leaning more towards the random assignment rule, especially in the Placement station, as the Keycat occurrences in the internal data are not considered. As we can see in this analysis, this has strong implications for the standard deviation of the workload between employees of a station. We can see in Figure 33a on the next page that the use of a random workload assignment at a station effectuates a very strong increase in expected standard deviation of the workload/utility between employees. No reduction in overall system performance is expected, with only an expected increase in total job system time of about 2.5% when the rule is applied to each station. A strong deviation of workload between employees, however, is not desirable and can in time lead to employees quitting. It is not only that there exists a constant discrepancy between workload between employees,

but also the amount of work a single employee has each day. It is therefore difficult to achieve a constant work rhythm, plan ahead, or make plans for after work as each day it is to be seen if overwork is required to finish the workload.
In conclusion we can state that changing the way workload is assigned to employees can have a significant impact on the deviation of workload between employees and the same employee on different days. If the amount a Keycat occurs in the data is not considered with the assignment of workload to different employees, we can strongly advise to do so. Afterwards, even more complicated rules might be constructed to encompass a wider range of variables.



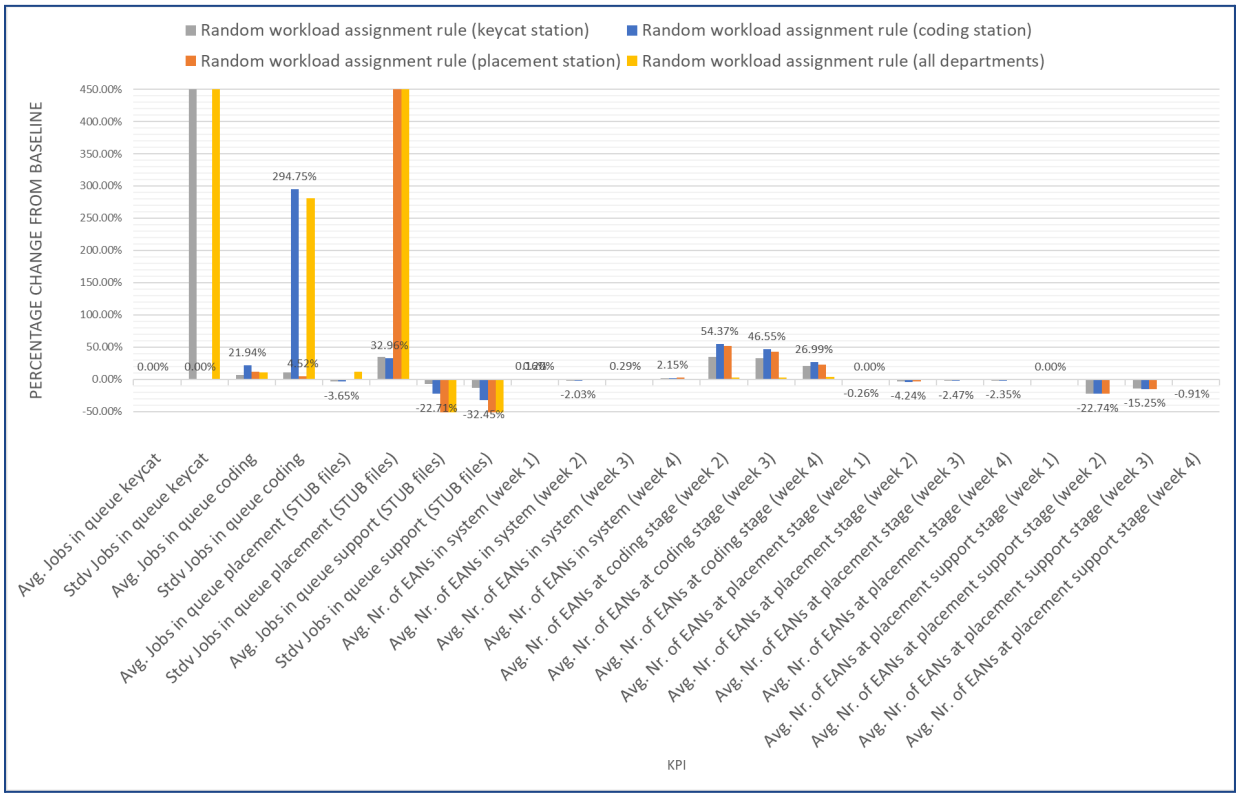


Figure 33a & 33b - Percentage change of using a random workload assignment rule

8.7 - Impact of increasing workload standard deviation and mean

In this section we will analyse the expected effects on system performance that increasing the standard deviation and/or the mean of the UPC arrival distribution has. A scenario such as this is expected after a new retailer introduction when the one time overflow workload is integrated (see Figure 14 of subsection 6.3.4). We do note that a situation where the mean of the arrival rate is increased without a decrease of the percentage eligible for Keycat is not very realistic (see Figure 16 of subsection 6.3.4) although the exact implication of such a situation requires additional research. For this scenario the eligible percentage is decreased from 92% to 75%.

We notice in Figure 34a an increase in EAN system time for every modification type. Also the number of EANs that could be submitted increases for every type, where the total change is highest for the types where the mean is also adjusted. Interestingly, multiplying the standard deviation by 3 without increasing the mean has the effect that 15% more EANs could be submitted compared to the baseline. Figure 34b shows an interesting result where, with an arrival rate standard deviation x3, most of the increase in standard deviation of workload between queues gets captured by the Keycat queue. Succeeding stations experience a significantly lower effect. Also, the Coding department catches the greatest increase in the number of jobs in the queue. This can also be seen in the increase in the number of jobs present at a stage per week, where the increase is highest in the Keycat and Coding stages. Also, with increasing the arrival rate standard deviation by 3 times, it is noteworthy that not the Keycat stage but the Coding stage has the greatest percentage increase in standard deviation of EANs present at this stage per week reaching almost 250% (not shown in Figure 34b).

We can conclude that the current system is somewhat resistant to an increasing mean and standard deviation of the arrival rate distribution, there is even room for submitting more EANs if more enter the system. This, generally, at the cost of an increase in mean and standard deviation of the workload for all departments. This would mean that the main problem that exists with a retailer introduction is how to handle the one time overflow workload. This is analysed in Section 8.12.

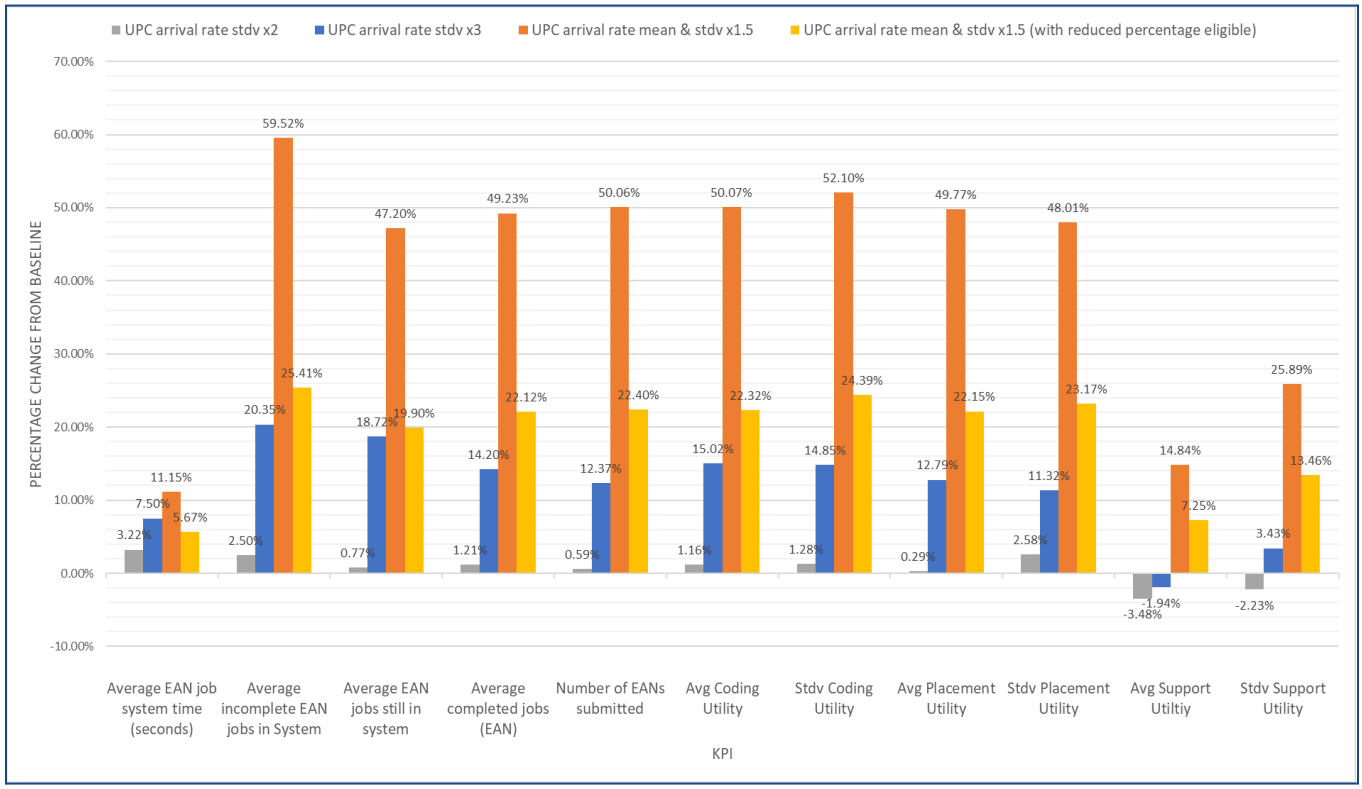


Figure 34a - Percentage change of increasing the mean and/or standard deviation of the arrival rate distribution

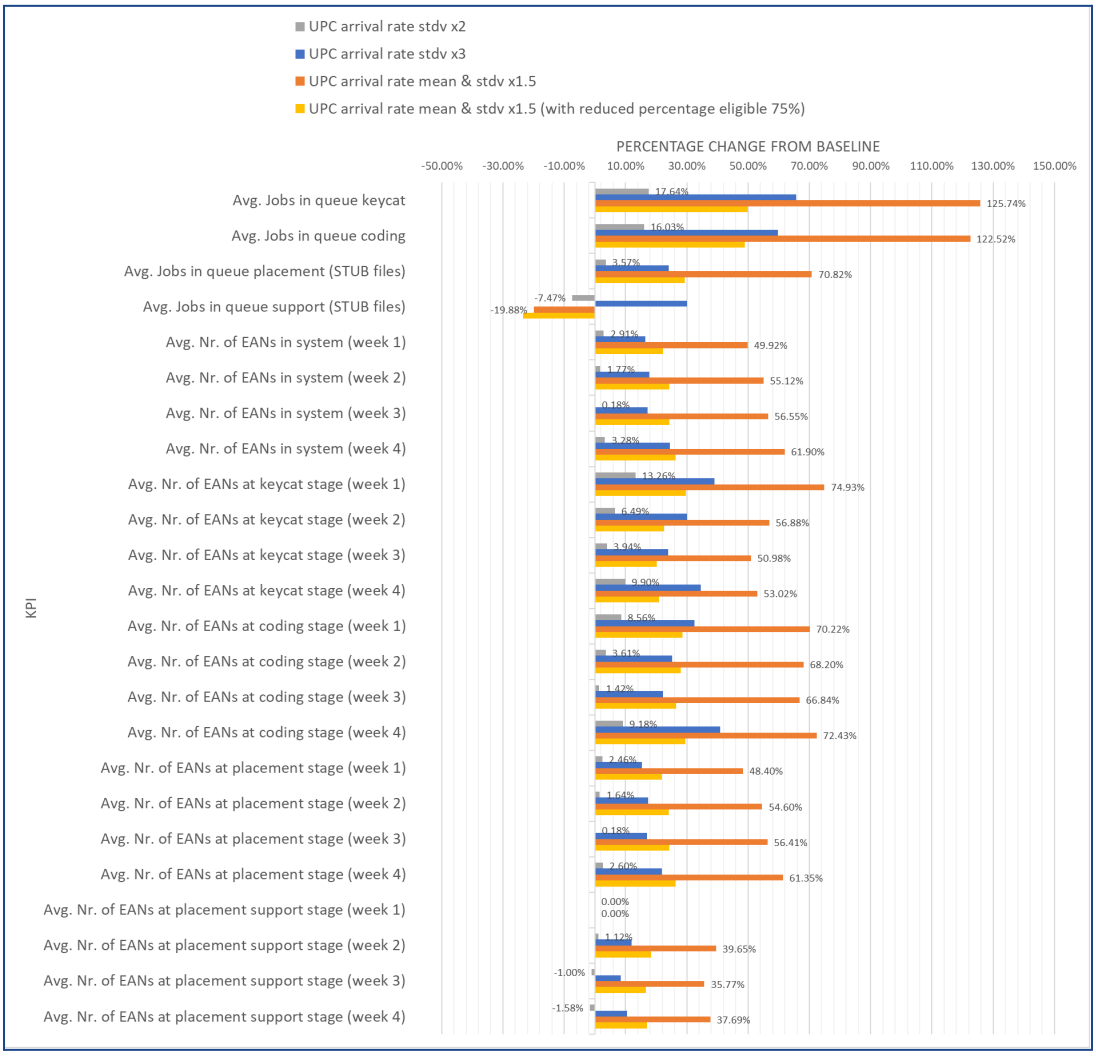


Figure 34b - Percentage change of increasing the mean and/or standard deviation of the arrival rate distribution

8.8 - Impact of increase of AUTO ENGINE Keycat percentages

This section will look at the effect modifying the AUTO ENGINE percentage has on the system. As we know, a percentage of the incoming jobs will be assigned with the right Keycat by an artificial intelligence engine automatically. This only happens when the engine is 100% sure, and this happens in about 30% of cases. This process alternative examines how the performance of the system changes if the AUTO ENGINE would automatically Keycat a greater amount of UPCs. To achieve a higher percentage in real life, investment in the accuracy of the AUTO ENGINE would be required. If the AUTO ENGINE is more often 100% sure, the percentage of automatic Keycats increases.

As Figure 35 on the next page shows, the system is expected to experience no real significant performance gains. The total job system time and the number of jobs that could be submitted remains about the same. The Coding department, where this change is expected to have the most effect, is not expected to experience a significant decrease in workload, although a 2.5% to 5% decrease might be worth it. A reason for these low percentage changes might be that the set days on which the Keycat and Coding tasks are performed are not changed. As there is less workload generated by the Keycatting task, more time can essentially be spent on the Coding task. When changing the days on which these tasks are performed to one day Keycatting (from 2), 3 days Coding (from 2) and 1 day rework, results of the simulation show Coding utility can be decreased to a percentage difference of the mean of -4.81% and standard deviation of -1.68% in the 60% modification

type. This does, therefore, not do a lot. Another effect we can see is that the percentage change of the standard deviation of the Placement utility decreases the same amount or even more than at the Coding department if an AUTO ENGINE percentage of 50% or higher is used In conclusion, improving the AUTO ENGINE to be able to Keycat a larger number of jobs automatically is expected to have some effect on workload and workload standard deviation in both the Coding and Placement departments.

We conclude that whether the investment in a better AUTO ENGINE is justified with these expected performance gains remains to be seen. Increasing the AUTO ENGINE Keycat percentage alone is not worth it according to us. This modification could be more advantageous in combination with other process modifications such as decreasing the workload of a separate Keycatting team such as in Section 8.5. Also, when the total workload increases significantly, like in the case of a new retailer integration, a higher automatic Keycat percentage could be beneficial as the Keycatting stage experiences the greatest workload increase (since it is the first department and not every item continues to the Coding stage).

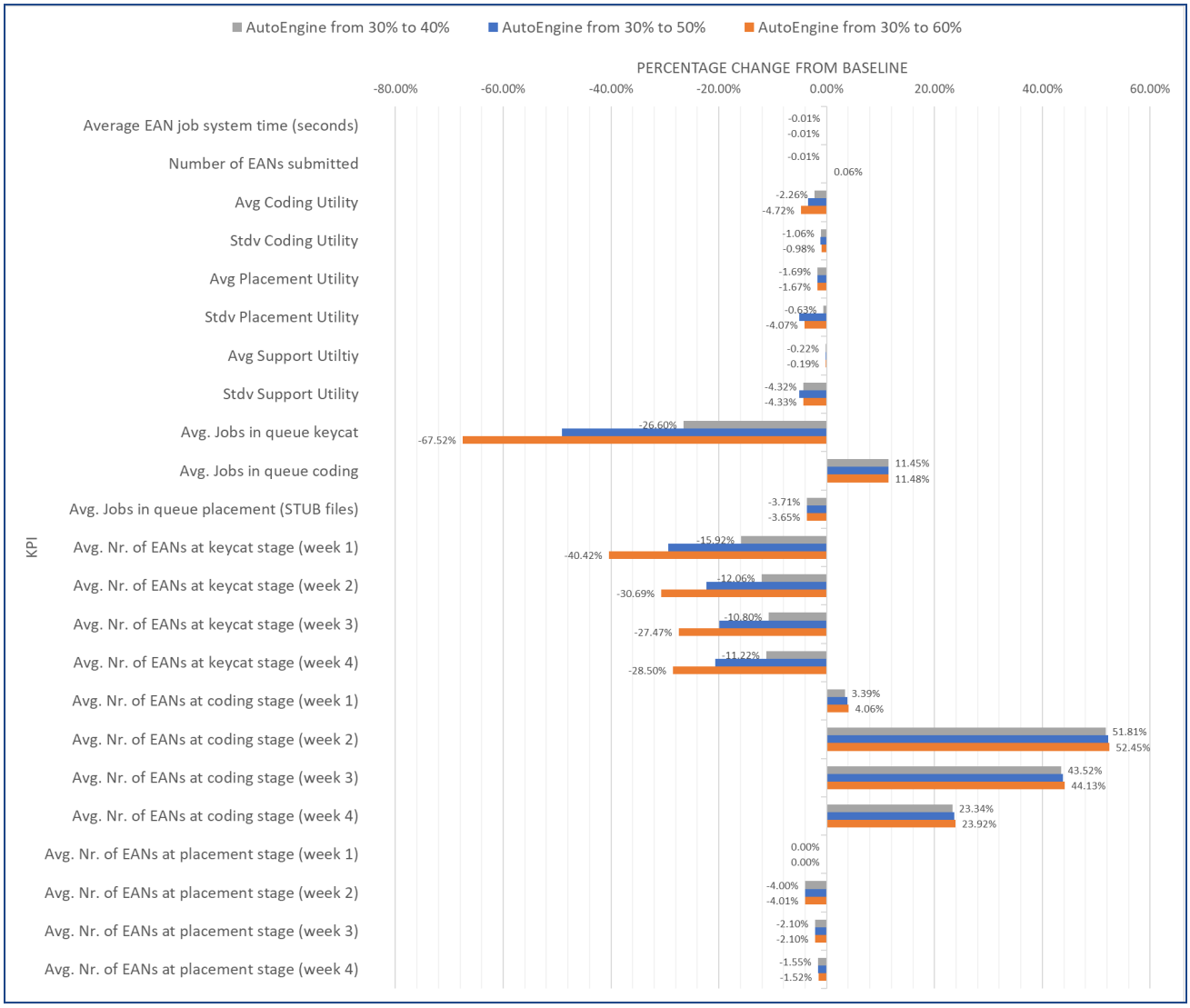


Figure 35 - Percentage change of increasing the AUTO ENGINE Keycat percentages

8.9 - Impact of changing the number of data arrival moments

In this section we will analyse the effect that the data arrival days are expected to have on the system performance. In the baseline scenario, data arrivals occur once a week before the first Keycatting day. It will be interesting to see if more data arrival moments with less jobs per arrival moment can have an impact on the system performance.

First we look at the effect of reducing the number of data arrival moments from 4 to 2, which means 1 data arrival moment per 2 weeks. Figure 36a shows quite negative expected percentage change of total job system time. A higher expected number of EANs have been submitted however. We do need to include a side note to this observation, as the simulation model stops after 4 weeks and shows the number of jobs still in the system, in the real system however EANs still present in the system at the end of one production period get processed in the next production period. We can indeed see a strong reduction in the number of EAN jobs that are still in the system in the simulation.

If data arrival moments are doubled, so 2 times per week, we can see overall positive expected results. Total job system time is expected to significantly decrease while the rest of the performance is expected to remain the same. This means that the data availability of submitted EANs can be brought forward, without any performance sacrifices or increased workload for employees. An even more extreme change to the data arrival moments, where each day new data arrives in the system, is expected to have reduced positive effects compared to doubling the number of data delivery moments. This can most notably be seen by the difference in percentage change of the total jobs system time (-12% for doubling the arrival moments versus -5% for every day data arrival).

We can conclude that increasing the number of days data arrives in the system from the data input department, can have a strong positive effect on turnaround time. However, an increase to daily data deliveries is not recommended, both because of the reduced expected performance gains compared to doubling the number of arrival moments and the real life practicality of such a process modification. For this process alternative we assume that each data arrival moment has the same arrival amount distribution.

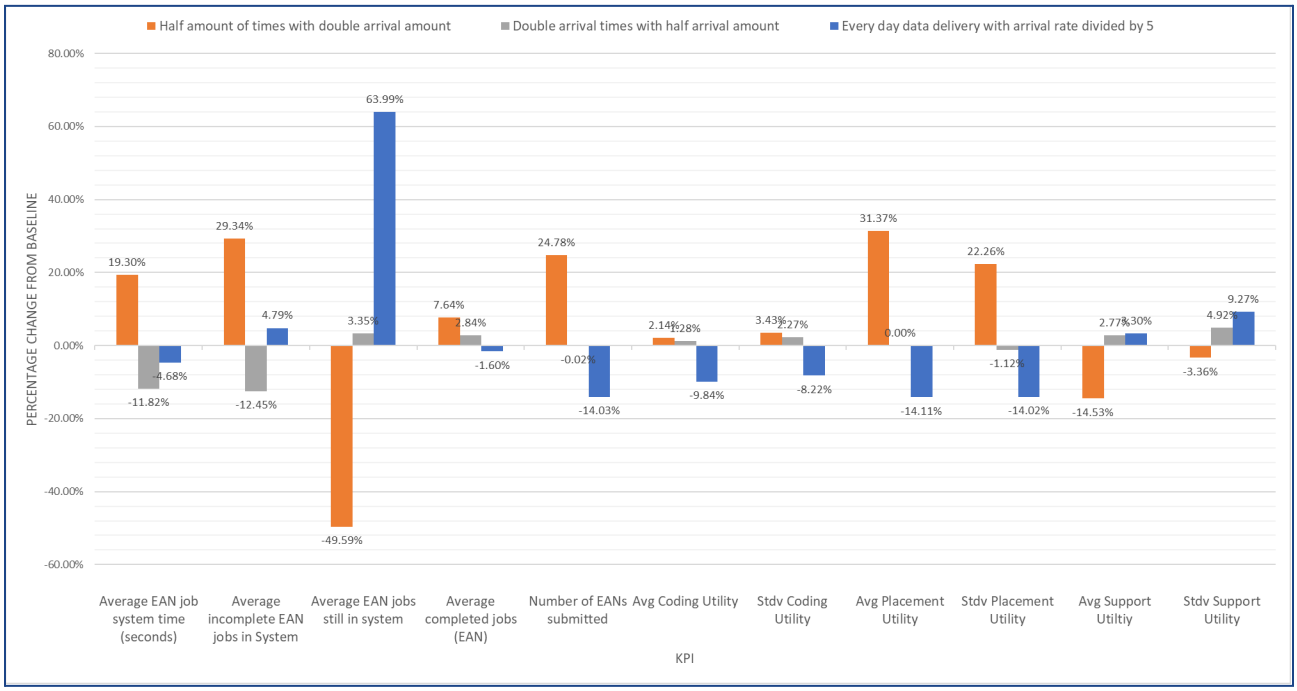


Figure 36a - Percentage change of changing the number of data delivery moments

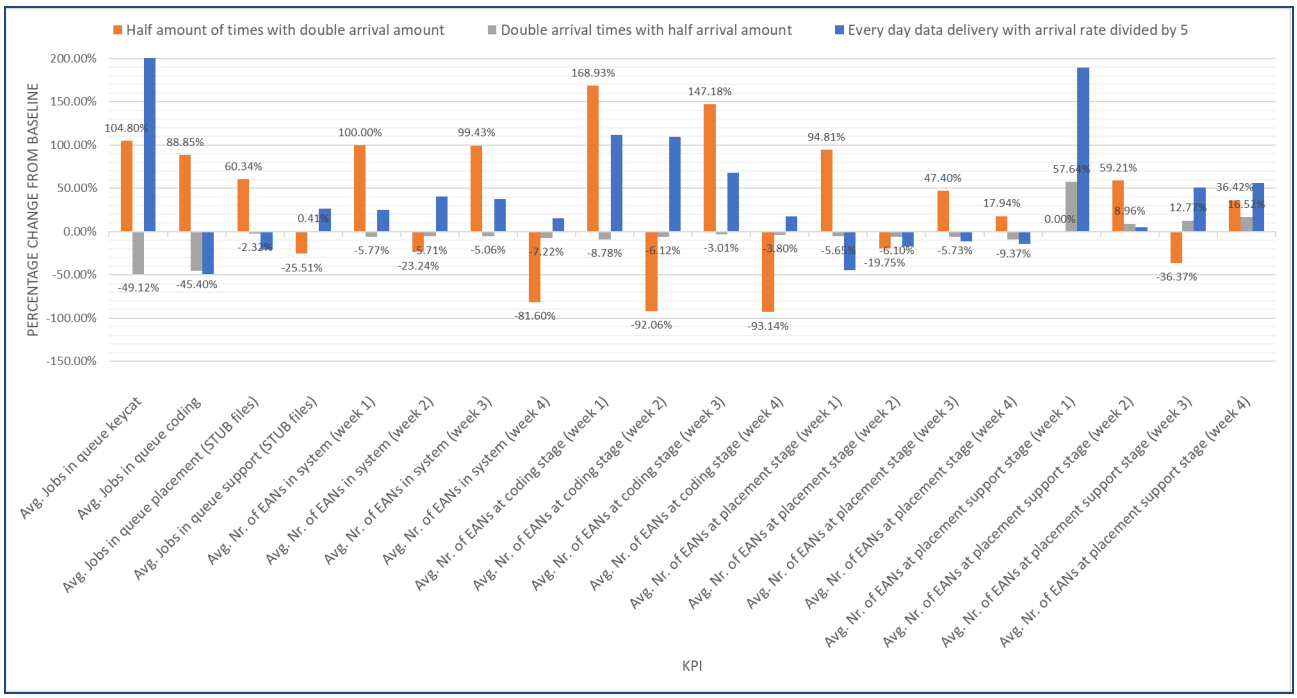


Figure 36b - Percentage change of changing the number of data delivery moments

8.10 - Keycat and Coding as one single task

In this section we will analyse the effect combining the Keycat and Coding station (and jobs) has on the performance of the system. This modification means that an employee immediately codes (assignes attributes to) the job that he/she just assigned to a Keycat in case that the Keycat is maintained. We also look if using a single queue, as analysed in Section 8.5, improves or deteriorates the situation.

Indeed we can see multiple interesting changes in the performance indicators shown in Figure 37a. Without the use of a single queue, a slight reduction (-2.5%) in total job system time is expected. Using a single workload queue, no reduction in job system time is expected, however the number of EANs that are submitted is expected to increase slightly (+1.7%). In contrast, without a single queue this number is expected to decrease by 1.6%. We also see that when combining these tasks the expected utility of the Coding department is increased, this increase in workload is less with the use of a single queue. The most significant percentage change occurs in the standard deviation of the workload that is expected to decrease a lot.

We can conclude that combining the Keycatting and Coding task is an interesting process modification to consider to reduce the workload standard deviation between Coding employees. We consider the expected performance improvements in either job system time or number of submitted EANs to be too low to justify implementing this process change. The practical value of this process alternative remains to be seen. Coding employees cannot first finish the Keycatting workload and afterwards focus on attribute Coding, but have to switch windows to perform the attribute Coding action in 40% of jobs.

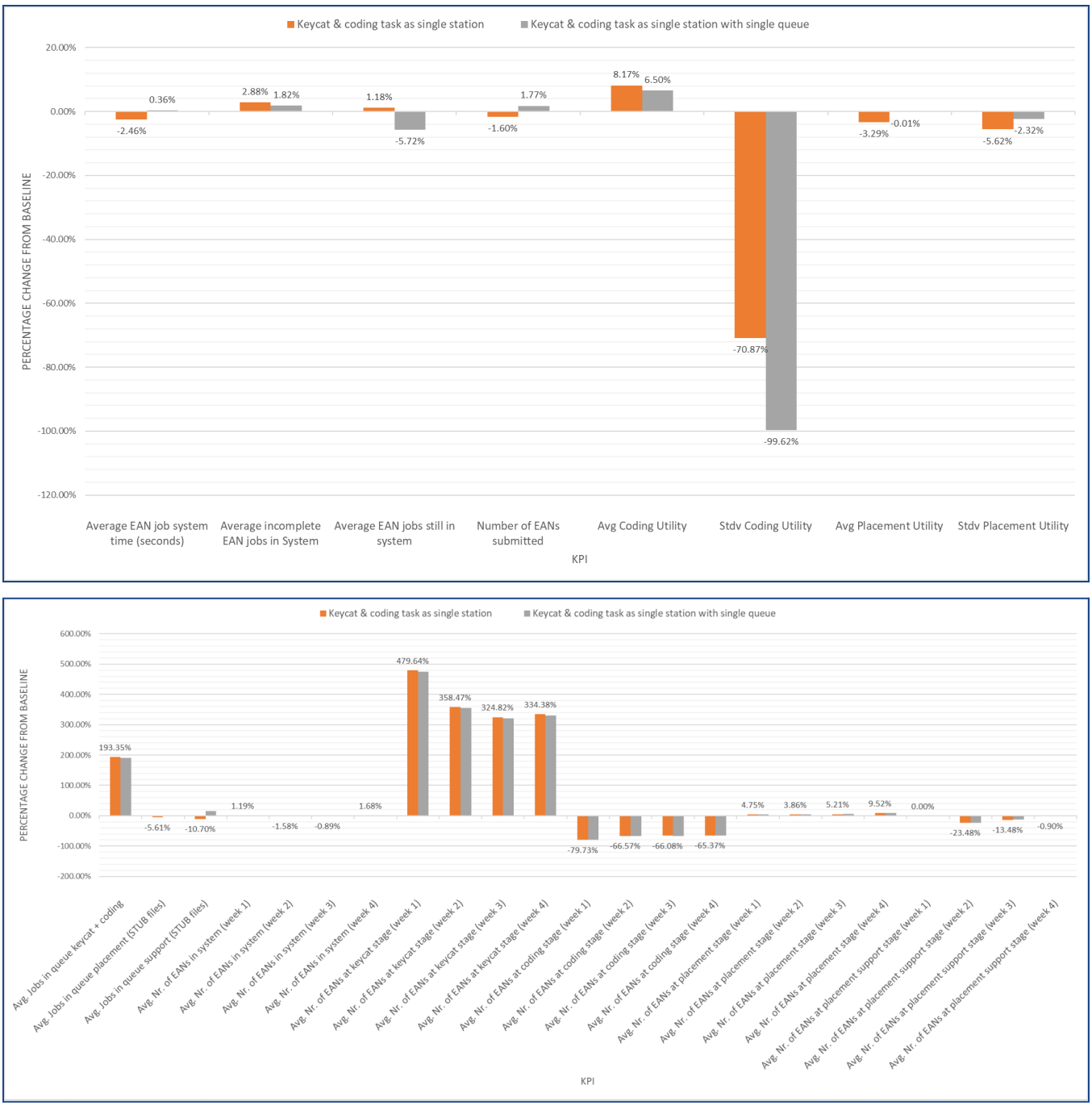


Figure 37a & 37b - Percentage change of combining the Keycat and Coding tasks

8.11 - Different system setups with bol.com integration scenario

This section will analyse the performance of multiple different process setups in the Bol.com data integration. One of the biggest operational questions within IRI is how to deal with the overflow workload typically experienced by the data integration of a new retailer. Often, there exists a significant discrepancy between products already present in the IRI mainframe and products sold by the new retailer. All these new products have to be processed and ensures this overflow workload as can be seen in subsection 6.3.1. For the analysis we use a constant UPC arrival mean and standard deviation with a decreased eligible percentage estimated based on the results shown in Figure 18 of subsection 6.4.2. We analysed 5 different setups where we compared the performance of the 5 setups to the baseline scenario and the performance of the last 4 setups compared to the first. For the analysis we assume that it is necessary to use a separate team responsible only for the data incoming from the new retailer in any case. This way the normal workflow will be minimally impacted. This means that in most cases new (temporary) employees have to be attracted. In such a scenario there exist three main questions:

- How many employees are necessary to handle the overflow workload
- Do the new employees have to be assigned to the team responsible for the normal workload or to processing the overflow workload
- What is the time that is expected to process all of the overflow workload

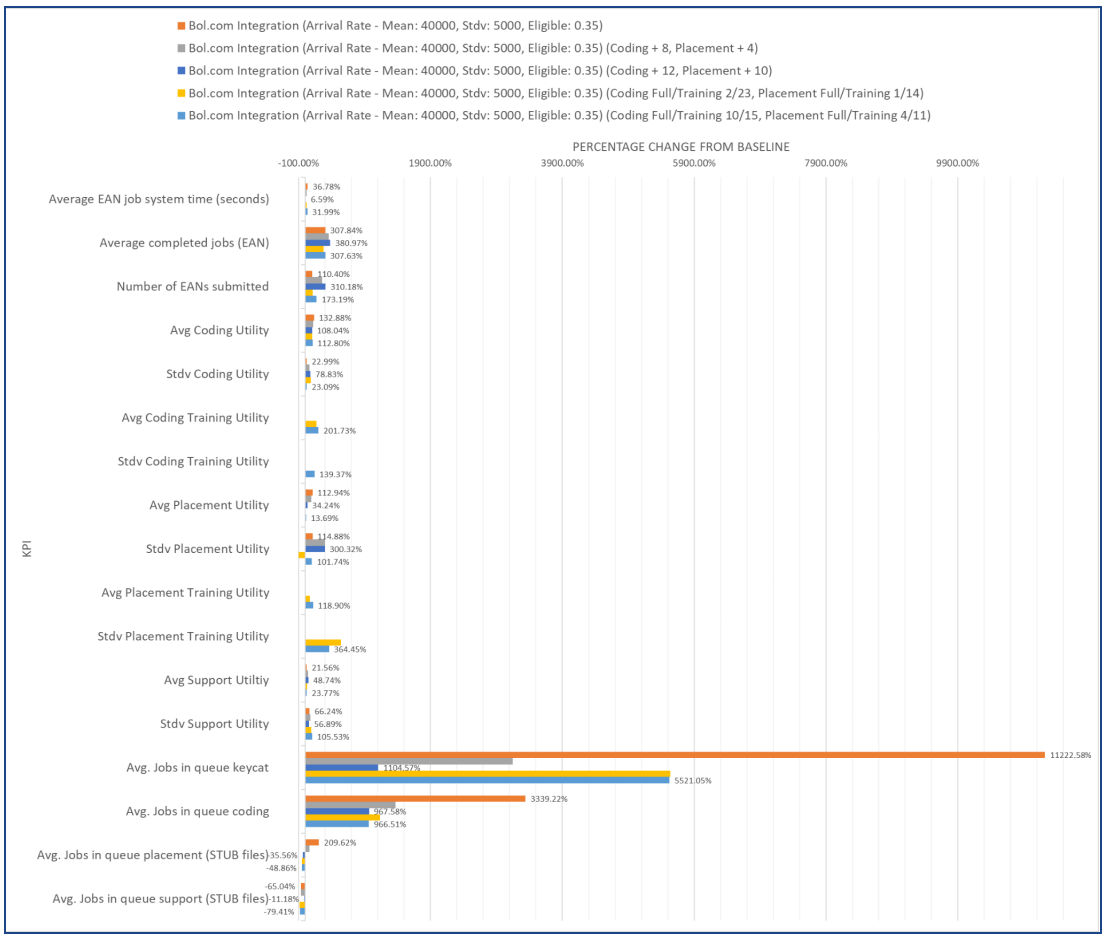
Although we try to give an answer to these questions in this section, the numbers remain an indication. We do however note that it is recommended to research this specific problem in more detail. We give management recommendations in chapter 9.

8.11.1 - Performance compared to the baseline scenario

As expected, the setup where the most full employees are included in the model performs the best. Figure 38a on the next page shows that this setup is expected to have the smallest increase in total job system time and the highest number of EANs that have been processed. Whether it pays off to assign such a high number of full employees to the overflow workload team does need to be considered. These employees will not be available to process the normal workload and new employees will have to be used in that team. Also, to achieve this number of full employees, it will be necessary to train new employees in advance.

The biggest effect an overflow workload has on the number of jobs in the system per stage is in the Keycatting stage (Figure 38b on the next page), especially with a baseline number of employees or with the inclusion of employees in training. Even though a Keycatting job requires less processing time than a Coding job it might be a good idea to set up a separate Keycatting team as defined in Section 8.5.

The increase in utility of both the Coding and Placement (mainly the first two setups) employees is not expected to impose significant problems. The utility of employees is quite low in the baseline scenario. This is because the model only considers workload directly related to the processing of EANs, even though real life workload consists of more activities than these. It also shows that in the chosen baseline scenario there is probably room for an increase in workload. The overflow workload team however, will have considerably less other activities than those directly related to the processing of EANs and therefore a significant increase in utility is not likely to impose problems. The only effect it might have if it is indeed too high and would imply the need for employees to use overwork hours, the expected number of jobs that can be performed in the 4 weeks we look at in the model will decrease. The relatively low utility of the Placement employees in the last 3 setups could imply the number of Placement employees could be decreased here to achieve a more consistent increase in utility.



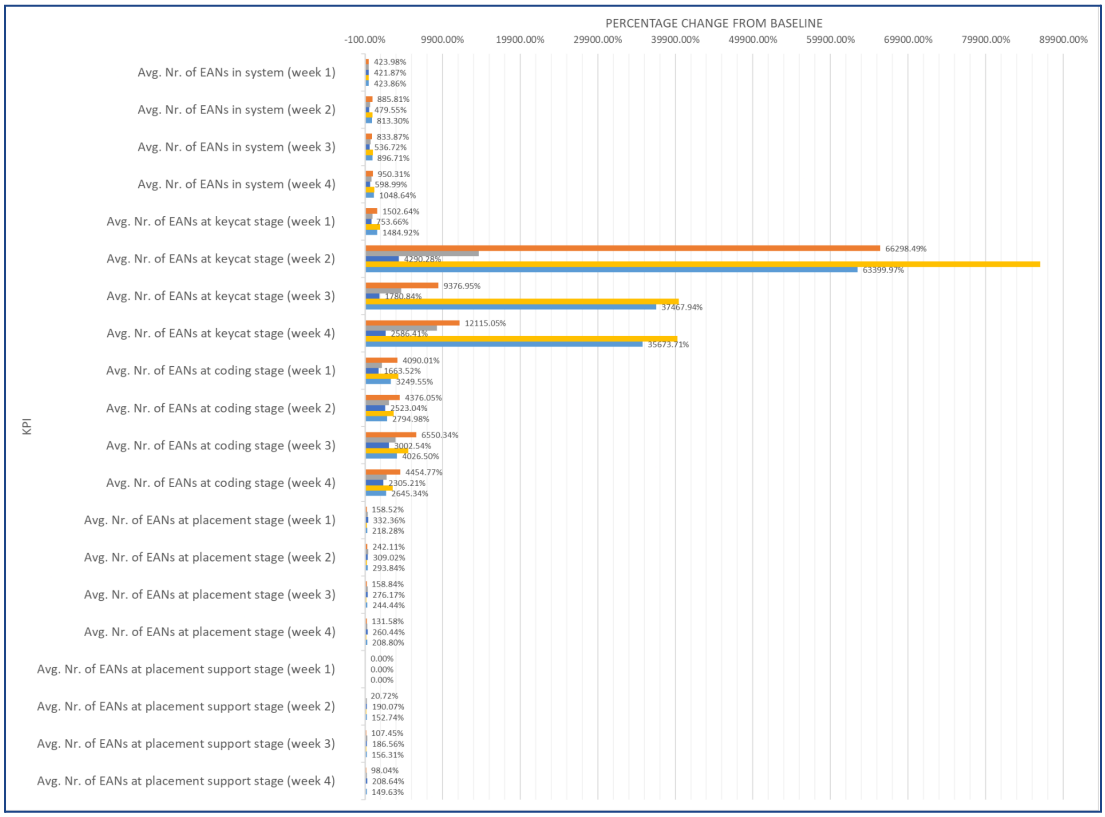


Figure 38a & 38b - Performance of five system setups under overflow workload

We now look at difference in the number of jobs that have been completed per setup, especially the number of EANs submitted, and the expectation of the number of EANs that need to be submitted if the whole overflow workload of the 4 weeks is to be completed. This gives a better indication of the added value of full employees over employees in training and an indication of the expected extra time needed to process all workload arrived in the 4 weeks of this scenario. The expected number of EANs that need to be submitted is 22400, which is 4 weeks of an average of 40000 arrivals per week where only 35% is eligible for a Keycat, and 60% of that amount is only assigned a Keycat and will not be submitted. The difference per setup is shown in Table 6:

Table 6 - Analysis of the number of EANs submitted

Composition teams Coding Full/Training: CF/CT Placement Full/ Training: PF/PT	1 CF: 15 PF: 5	2 CF: 21 PF: 9	3 CF: 25 PF: 15	4 CF/CT: 2/23 PF/PT: 1/14	5 CF/CT: 10/15 PF/PT: 4/11
Number EANs submitted	6017.96	10152.71	11732.25	6177.7	7813.83
Percentage from total EANs to be submitted	27%	45%	52%	28%	35%
Weeks to submit all	14.89	8.83	7.64	14.50	11.47

From table 6 we can get a sense of the expected amount of time that is needed to process all the workload associated with the integration of a new retailer (overflow workload). The expected number of EANs that in total need to be submitted to complete all overflow workload is compared with the number of EANs that have been submitted in one production period according to the simulation model. We compare 5 different team compositions. As shown in Table 6, the highest number of submissions is obtained in setup 3, which is again not surpriseable. Doubling the number of Coding employees and increasing the number of Placement employees by a factor 3 almost doubles the expected number of submissions. Interestingly, a team composition with a high number of employees in training achieves only a slightly higher number (~160) of submissions as a composition with half the number of full employees. This would make a case to include as many full employees in the overflow workload team, as we have seen that including employees in training in the baseline workload team has significantly less impact on total performance.

8.11.2 - Performance compared to the setup with baseline employee numbers - setup 1

In this subsection we look at the performance of system setup 2 to 5 compared to the performance of setup 1. This will give an idea of the performance gain of adding either full or new employees to the overflow workload team compared to the amount defined for the baseline scenario. We again see that the addition of new employees in this team negatively affects system performance compared to having only full employees (Figure 39 on the next page). The stage of the production process that sees the biggest increase in expected number of jobs per week when including new employees in the model is the Keycatting stage. This effect could be mitigated by a range of different solutions. Again, a separate Keycatting team could be set up, the AUTO ENGINE could be improved to Keycat a higher percentage of jobs or the retailer to be integrated could be asked for additional data besides the transaction amount and EAN number so Keycat assignment is easier.

We conclude that the assignment of full employees is inherently better than new employees for the processing of overflow workload, especially if the time this workload needs to be submitted is limited. However, if not enough (extra) full employees are available they would have to be trained, which adds to the total time necessary to handle the

workload. The training could however be done before the arrival of the first data of the new retailer. If new employees are used, it is advisable to create a team with new employees only and one or two full employees for oversight. This will be at the expense of the total number of EANs submitted per period. Given these results are generated without the use of possible process improvements discussed in the previous sections of this chapter.

It must be noted that this is a very complex issue, and that this analysis is based on only 1 retailer introduction while the model disregards more in-depth process characteristics that might influence the workings in such a scenario. Also other factors related to retailer introductions and overflow workload have not been investigated yet. Recommendations on next steps will be outlined in chapter 9.

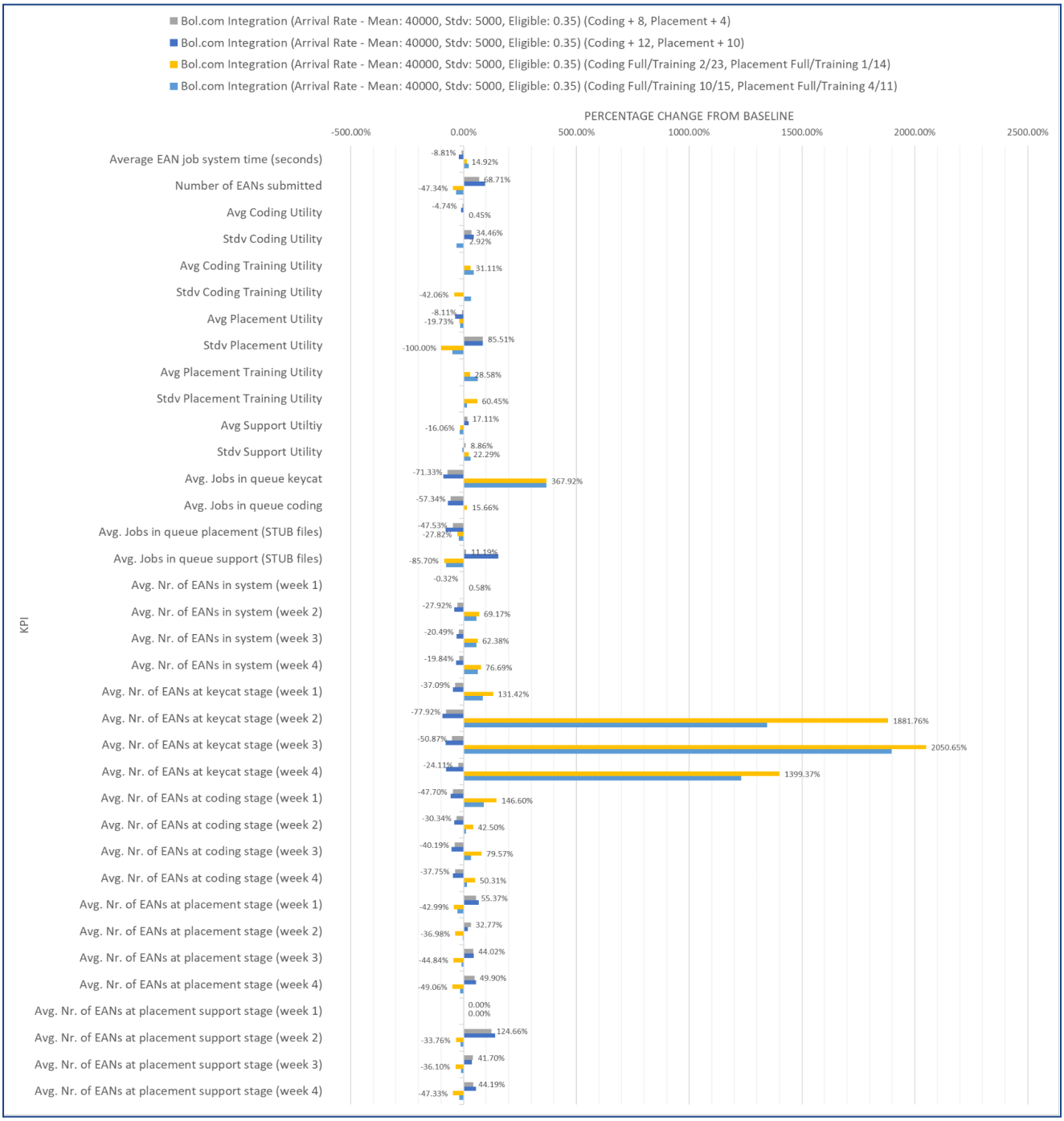


Figure 39 - Performance of system setup 2-5 with setup 1 under overflow workload

9 - RECOMMENDATIONS, DISCUSSION & CONCLUSION

In this chapter we will briefly summarise the steps taken to answer the research question and solve the core problem of this research. The results and analysis of the process alternatives of chapter 8 are then used to provide a range of recommendations to the company, both concerning the most interesting process modifications as areas of follow-up research. We will then provide a discussion on the taken approach, the implemented model and the results to provide the reader with a nuanced insight into the choices made and areas where the research can be improved. We end with a general conclusion.

9.1 - Research summary

This research was performed at Information Resources Inc., which is a market researcher and data provider in the Fast Moving Consumer Goods sector. With the commission of this research the management of IRI intended to gain a different, Operations Research centred, look at the internal processes of the company (in particular the processes of the operational department). This department experiences many challenges, does not function as management would like and lacks efficiency. Although IRI is a company where many problems are identified and tackled, in this case a more in-depth study is needed. As there is no Operations Research/Industrial Engineering expertise present at IRI, this research was started from scratch without a predefined assignment present. A set of core problems are defined based on a cause-effect analysis of the problems & opportunities (Figure 4 in Section 2.1) present at the company. The core problem with the highest added value in solving based on an analysis of key processes and operational challenges (Section 2.2 & 2.3) served as basis for the research question defined for this research (chapter 3):

Can a Discrete Event Simulation (DES) implemented in general-purpose programming language provide relevant quantitative information on process performance in the changing stochastic operational environment and provide management with the necessary insight to make better substantiated strategic decisions?

With this research question we verify if the application of a quantitative stochastic Operations Research model (Discrete Event Simulation - see Chapter 4) of the operational process at IRI can clarify the abstract and complex Operational processes and provide quantitative insight in these processes to make better and founded strategic decisions and changes to the process. To provide an answer to this research question we started with a literature review of the research domain of stochastic production processes that aims to:

- Provide substantiation why DES is the most suitable stochastic modelling technique to apply (Section 4.1)
- Provide information about the characteristics, evolution and latest developments of the DES technique and where the research is expected to add to existing research (Section 4.2)
- Provide substantiation for the DES modelling approach: the consideration between DES software and a general-purpose programming language implementation (Section 4.3)

To gain a more concrete and in-depth insight into the complete operational process, we have constructed a Business Process Model of all activities that are performed per department (Figure 7 of Section 5.1). We then defined the key stochastic parameters present in the operational process (Section 5.2) and classified the operational process of IRI as an intermittent batch production process (Section 5.3). We have made the decision to focus exclusively on the processes of the Coding and Database Analysis (DBA) departments to decrease the complexity of the model, this choice is substantiated in Section 5.4. A queueing network of these two departments is constructed to serve as reference for the DES implementation (Section 5.5).

We have collected historical internal data, which we analysed with Python to determine the following input for the DES model:

- The time period to serve as stable baseline workload scenario (subsection 6.3.2)
- The best fitting theoretical distribution for arriving Universal Product Codes (UPCs) in the baseline workload scenario, which is the Normal Distribution (subsection 6.3.2)
- The amount of workload and the order of processing of different types of jobs at each stage, defined as percentage of total arriving UPCs.
- An analysis of the characteristics of the workload amount associated with a new retailer integration (Section 6.4)
- The occurrence of product categories (Keycats) in the data (some Keycats occur more often than others) (Section 6.5)
- The best fitting theoretical distribution functions of task processing times (Section 6.6), where we have made informed assumptions about tasks without available data (DBA process: subsection 6.6.2, employees in training: subsection 6.6.3)
- The number of times the client provides feedback on performed work, which results in rework (Section 6.7)

In chapter 7 we provide an overview of the steps that we have taken to implement the processes of Coding and DBA into a DES application. First we substantiate the choice of using Kotlin as general-purpose programming language (Section 7.1 and the Java Simulation Library as open-source DES library (Section 7.2) to decrease development time. We then made an abstraction of the Coding and DBA processes in the form of a conceptual model (Section 7.3). This conceptual model defines the elements the DES Kotlin implementation consists of. The simulation horizon is set as a finite period of 4 workweeks (1 production period) and the simulation precision as 1 second (subsection 7.4.1). We define a preset number of 100 replications, which aims to find a balance between running time and solution deviation and also serves to reduce the complexity of using the model in practice (subsection 7.4.2). The remaining sections of chapter 7 contain in-depth implementation details.

Chapter 8 contains the results of a single-effect analysis of the expected value of a set of 11 process alternatives. The most promising results are used to make a number of recommendations to the company to both improve efficiency of the processes as to define promising domains of follow-up research. The percentage difference of the performance of a process alternative compared to the baseline scenario is used to determine the value of a process modification. To give the reader an idea of the results obtained we have included three with the highest expected added value below.

- A workload assignment rule that assigns workload to employees by ordering the jobs based on the occurrence of product categories in historical data could decrease workload deviation between employees by 50% to 75%.
- Setting up a separate team that handles the assignment of new product codes to a product category could increase total jobs submitted by about 2% without affecting the performance of the rest of the system.
- Increasing the “raw” data delivery moments from one a week to 2 times a week, smoothing out the arriving workload could decrease the throughput time by about 10% while decreasing the workload of employees by 10%.

Additional recommendations and insights based on the results of this research can be found in section 9.2.

9.2 - Recommendations

This section will summarise the results of chapter 8 and provide recommended actions for the management of IRI to take. We have ordered the recommendations based on expected added value and ease of implementation. The recommended actions based on the results of this research presented below can be used in two ways:

- Process modifications that affect performance indicators in a desirable way can be directly implemented
- The results of chapter 8 and the recommendations can be used to define the areas or modifications of the operational process were follow-up research is most relevant

Besides the recommendations presented in this section and the results presented in chapter 8, the in-depth analysis of the operational process of chapter 5 and the DES application can be used to analyse other process alternatives that might become of interest in the future. Follow-up research in the operational process of IRI can use this research as a foundation and findings can be added to the DES application to improve the model accuracy and analyse the effect of the findings. The main benefits of the recommendations on the efficiency of the operational process will be a decrease in total job throughput, an increase in number of EANs that are submitted in a production period and decreasing the deviation of the workload between employees. In addition, recommendations are made to more effectively incorporate new employees into the operational departments. We also provide ideas on a more efficient incorporation of new retailers.

9.2.1 - Changing/implementing the workload assignment rule

One of the modifications that has the most effect on employee workload standard deviation is the way workload is assigned to employees. Workload standard deviation between employees comes up again and again as a very significant problem for employees in the operational department. The implementation of a simple ordering rule of jobs based on their Keycat (instead of random or based on client), is expected to have a very positive effect on employee workload standard deviation (especially in the Placement stage). When workload is assigned randomly instead of considering the amount a Keycat occurs in historical data, as is the case at the moment, increases the standard deviation of employee workload by 200% at the Coding stage and more than 500% at the Placement stage. Making sure workload is divided equally considering historical Keycat occurrence is therefore recommended.
Follow-up research could focus on a more advanced workload assignment rule/algorithm. A Linear Programming model could be used to determine the optimal assignment of workload based on the employee performance and continuously updated data. The moments the assignment should be changed could be considered in the model. As a situation evolves (changing period of the year, changing market conditions, changing employees) it might be valuable to adjust the assignment, however this should not be done too often as this involves negatives such as temporary reduced efficiency of employees and adjustment to the new assignment rule.

9.2.2 - Separate Keycat team or Keycat and Coding as single task

Separating the Keycat and Coding tasks into two separate teams is expected to increase the number of EANs submitted in a production period and can also positively impact the standard deviation of the workload in the Coding stage. This process modification has the advantage that both the Keycat task as well as the Coding task can be performed every day instead of one day on one day off. It would depend on the workload, but with baseline workload, transferring 2 or 3 Coding employees to the separate Keycat team would be advised. Results of the simulation model also show very positive results when combining this modification with an increase in the AUTO ENGINE automatic Keycatting percentage and the number of data arrival moments. This would combine an expected reduction in total job system time of 20%, an increase in EANs submitted of about 10% and a reduction of workload deviation.

Combining the Keycat and Coding task in a single task is expected to be less practical than using a separate Keycatting team. The simulation model shows similar effects as a separate Keycatting team and further shows a significant reduction (~70%) of the standard deviation of the Coding workload. The settings used might need to be better tuned to the real situation, as the fact that a combined Keycatting and Coding task would require constantly switching between screens in the program that is used for these tasks, processing times might be longer because of this but also because an employee cannot focus on a single task.

9.2.3 - Changing the number of data arrival moments

The simulation model shows a significant positive effect on the total job system time and data availability (~10%) if the process is redesigned to allow for more (smaller, 2x per week) data arrival moments (by redesigning the data input process). It is also expected that it would reduce the Coding and Placement workload by about the same percentage. An even greater increase to every day data delivery moments is not recommended based on results of the simulation. Decreasing the number of data arrival moments is also not recommended, this modification is only expected to have a positive effect on total EANs submitted at the cost of total job system time and workload of employees. Other changes to the IT system such as increasing the AUTO ENGINE automatic Keycatting percentage is not expected to have significant effects on performance. It could, however, be considered in combination with other process modifications, especially if Keycat workload needs to be decreased.

9.2.4 - Changing the number of full employees per station

Changing the number of Coding employees is not expected to really impact overall performance of the system, only impacting the Coding utility which is expected. In contrast, modifying the number of Placement employees has a high impact on overall system performance. Increasing the number with 2 (from the 5 employees of the baseline scenario) can reduce total job system time by almost 6%, while a reduction is not recommended (in a scenario with comparable workload). Modifying the number of Placement support employees is another option. Their utility is significantly lower than the employees at other stages of the process. This is partly due to the fact that a significant amount of responsibilities of these employees have been left out of the model as they did not directly contribute to EAN submission. Management could consider a more direct separation of responsibilities for these employees. Also, a part of this team could be used to help out the Placement team when necessary. This way no employees have to be trained especially.

9.2.5 - Effects of employees in training

Section 8.3 shows that the inclusion of training employees has quite a negative impact on various aspects of process performance. This is especially true for when full employees are exchanged with new employees, just an addition of new employees to an existing team of full employees reduces overall utility slightly. The negative effects can however be decreased when considering the results of section 8.3. We have noticed that the exchange of employee types has the least negative impact on the Coding department. This could be used to IRIs advantage. A new employee at IRI needs to be trained in two areas. They need to learn to perform their responsibilities/tasks but they also need to get familiar with the (quite complex) company. This consists of generic knowledge like the responsibilities of different departments, the meaning of abbreviations and concepts used within the company, the company structure, responsibilities of colleagues etc. Some of the negative effects of training new employees could be avoided by always training new employees in the Coding department. Fully trained Coding employees can be transferred to the Placement department when necessary. These two departments have a lot in common, in fact they were one and the same department a while ago. This way, the employees transferred from the Coding department will have learnt the Coding task (making learning the Placement task easier) and are already familiar with the company. This means trained employees transferred from the Coding

department affect the performance of the Placement team by only half or less than a new employee. This way, per employee about 2.5% increase in total job system time and 2% decrease in submitted EANs can be saved. Additionally, we expect that these employees are able to handle more workload.
Follow-up research could further investigate the performance of employees in training. The basic assumption to model the processing time of employees in training used in this research (fixed percentage decrease of mean and standard deviation) can be improved in many ways. The real difference in performance between full and new employees can be measured. The gradual increase in productivity and the effect new employees have on existing employees can also be investigated.

9.2.6 - New retailer integration based on Bol.com data

The question of what to do in the case of a big new retailer introduction is very difficult and definitely requires additional research only focussing on this problem. The fact that only data from 1 new retailer introduction is used requires the assumption that all new retailer introductions are comparable to the Bol.com integration. This might not be true. Also, we assume that we know precisely what the workload is (which we do since we use historical data), however this is not the case with a new retailer introduction. We do however recommend to make sure a separate team is set up to handle the overflow workload related to a new retailer, so the normal workflow does not get impacted. Following the results from the simulation model, this team would benefit from the highest percentage of full employees possible, if the goal is to integrate as fast as possible. Although not explicitly analysed in this research, Section 8.9 would suggest that increasing the number of data arrival moments could help with evening out the workload.
Follow-up research could focus on finding a way to determine the expected distribution of the workload with more precision. If more data can be found on historical introductions, also a more precise matching algorithm could be constructed that determined what part of the data of the new retailer does not already exist in the IRI mainframe and must therefore be processed (if eligible). One could think of scraping a random sample set of products from the new retailers website that can be compared with the data already available at IRI. The size of this sample set should of course be determined.

9.3 - Discussion

This section will provide a discussion on some of the elements of this research, this will provide the reader with information about important considerations made, the positive elements and the elements requiring additional attention.
Being the first research into the operational process of IRI combined with the high complexity of the complete process makes it impossible to model the process without making simplifications or assumptions. Important simplification steps that have been made are the focus on only two bottleneck departments, only considering workload directly related to EAN processing and the finite simulation horizon of one production period. These steps inherently affect the possibility to compare some of the results to their values in real life to allow for validation of the model for instance. This is why extra attention has been paid to the verification of the model by process experts within IRI. Additionally, the use of an object-oriented programming language has allowed the elements of the process that are included in the model to have a high resemblance to the real process (names, workload flow, process details). The extensive analysis and use of historical internal data, although common in simulation studies, also improves the resemblance of the model to reality and its credibility. We believe that setting up the experiments in a way where the model output for the baseline scenario and the model output of the process alternative are compared, is an effective way to determine the expected added value of process modifications.
What this research still lacks is a way to make better decisions in a changing environment. Many of the analyses in this research have been done based on historical internal data. The simulation model can also be used to analyse scenarios

that are more of a predictive nature. Without research that aims to substantiate the input parameters that are used in such a scenario the added value remains questionable. Separate research that has a particular focus on prediction rather than explanation could provide additional value for the aforementioned changing environment IRI finds itself in. New retailer introductions are expected to become more common, and the legacy way of providing data makes way for a more advanced online on demand manner with the use of the Unify platform and the change to the new Liquid Data technology. This changes the way of working for many departments, although the Coding and DBA departments are expected to be minimally impacted. We have already made some recommendations for follow-up research with a more predictive aim in the area of new retailer introductions, however, we believe that workload/Keycat forecasting or workload balancing solutions could also provide significant value for the operational process at IRI. These steps could in turn make the process more robust and allow for more disrupting process changes to increase data availability even further, which is the long term goal of management.

The modularity of the DES computer application makes it possible to incorporate new research into the domains indicated in Section 9.2 easily in the program, which will increase the accuracy of its results. The practical value of the model can be increased with some additional work on creating an intuitive GUI so the application can be used by management to perform process comparisons theirselves. The high complexity and digital/human factors present in the operational process of IRI make it unrealistic to digitise it completely, nevertheless we believe that this first version of the model represents a balanced simplification of the processes without sacrificing the useability of the results. We can conclude that this Kotlin DES implementation is able to provide quantitative strategic insight in the stochastic operational process. Management can make better informed and quantitatively substantiated decisions, both with using the results/recommendations and the practical use of the DES application. The research was therefore able to provide a positive answer to the research question.

9.4 - Conclusion

We conclude that the digitisation of the operational process of IRI into a DES application is able to highlight elements and characteristics of the real process that are difficult or impossible to determine without. The results and the analysis of the process alternatives can be used to make strategic changes to the process. Because of the comparison technique used where we compare an alternative to a baseline process setup, the performance differences are expected to also hold in the real process. Besides the substantiated strategic process changes the results and analysis of this research have made possible, it is now also possible to better understand the operational process and determine where additional research can have the biggest impact. New information gained from follow-up research can in turn be implemented in the simulation model to improve the accuracy. In time, the integration of multiple follow-up research time could allow for more in-depth process adjustments or optimisation of certain process elements. IRI now has a first foundation from which new steps can be taken with more ease.

This research adds to the existing research in the fields of DES and stochastic production systems modelling/analysis as well. This is the first research where Kotlin has been used to implement a real life production process as a DES model. Research containing advanced and complete implementations of a process with the use of the Java Simulation Library is also uncommon. Papers that consider open-source Java simulation libraries generally introduce new libraries and are from the author of the library. Limited research focuses on the implementation of an existing open-source library. The type of process that is modelled using DES in this research can also be considered as innovative. Plenty of research exists on the use of DES to model classical production systems or services processes, however research on modelling a mostly digital process with digital entities is new as well. This research can furthermore be used as a framework for the development of effective and practical DES applications for production system analysis using open-source DES libraries for general-purpose programming languages.

As stated in section 8.3, we believe that the goal of this research has been achieved. The operational process that was quite abstract, complex and undocumented prior to this research is cleared up and formalized. The main research question could be answered in this research. We have been able to develop a DES application with the use of Kotlin that has provided us with quantitative data on the operational process which has formed the basis of both the proposed process modifications as areas for follow-up research. The analysis results, as well as the DES application can be used by management to make substantiated strategic decisions to improve the efficiency of the operational process.

10 - REFERENCES

Section 2

Vieira, G. E., Kück, M., Frazzon, E., & Freitag, M. (2017). Evaluating the robustness of production schedules using discrete-event simulation. *IFAC-PapersOnLine*, *50*(1), 7953-7958.

Andersson, E. V. (2014). *Assessment of robustness in railway traffic timetables* (Doctoral dissertation, Linköping University Electronic Press).

Salido, M. A., Barber, F., & Ingolotti, L. (2008). Robustness in railway transportation scheduling. In *2008 7th World Congress on Intelligent Control and Automation* (pp. 2880-2885). IEEE.

Policella, N. (2005). Scheduling with uncertainty: a proactive approach using partial order schedules. *AI Communications*, *18*(2), 165-167.

Takeuchi, Y., & Tomii, N. (2005). Robustness indices for train rescheduling. In *CDROM Proceedings of the 1st International Seminar on Railway Operations Modelling and Analysis. Delft, the Netherlands*.

Section 4

Graves, S. C. (2011). Uncertainty and production planning. In *Planning production and inventories in the extended enterprise* (pp. 83-101). Springer, New York, NY.

Mula, J., Poler, R., García-Sabater, J. P., & Lario, F. C. (2006). Models for production planning under uncertainty: A review. *International journal of production economics*, *103*(1), 271-285.

Martinez, P., & Ahmad, R. (2021). Quantifying the Impact of Inspection Processes on Production Lines through Stochastic Discrete-Event Simulation Modeling. *Modelling*, *2*(4), 406-424.

Tako, A. A., & Robinson, S. (2012). The application of discrete event simulation and system dynamics in the logistics and supply chain context. *Decision support systems*, *52*(4), 802-815.

Winston, W. L., & Goldberg, J. B. (2004). *Operations research: applications and algorithms* (Vol. 3). Belmont: Thomson Brooks/Cole.

Kampa, A., Gołda, G., & Paprocka, I. (2017). Discrete event simulation method as a tool for improvement of manufacturing systems. Computers, 6(1), 10.

Robinson, S. (2005). Discrete-event simulation: from the pioneers to the present, what next?. *Journal of the Operational Research Society*, *56*(6), 619-629.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management science*, *1*(3-4), 197-206.

Dyer, M., & Stougie, L. (2006). Computational complexity of stochastic programming problems. *mathematical programming*, *106*(3), 423-432.

Bellman, R. E. (1957). Dynamic programming, ser. *Cambridge Studies in Speech Science and Communication. Princeton University Press, Princeton*.

Xu, S., Panwar, S. S., Kodialam, M., & Lakshman, T. V. (2020). Deep neural network approximated dynamic programming for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 02, pp. 1684-1691).

Babulak, E., & Wang, M. (2010). Discrete event simulation. *Aitor Goti (Hg.): Discrete Event Simulations. Rijeka, Kroatien: Sciyo*, 1.

Turner, C. J., Hutabarat, W., Oyekan, J., & Tiwari, A. (2016). Discrete event simulation and virtual reality use in industry: new opportunities and future trends. *IEEE Transactions on Human-Machine Systems*, *46*(6), 882-894.

Kádár, B., Pfeiffer, A., & Monostori, L. (2004). Discrete event simulation for supporting production planning and scheduling decisions in digital factories. In *Proceedings of the 37th CIRP international seminar on manufacturing systems* (pp. 444-448).

Alves, R. G., Maia, R. F., & Lima, F. (2022). Discrete-event simulation of an irrigation system using Internet of Things. *IEEE Latin America Transactions*, *100*..

Tavakoli, S., Mousavi, A., & Komashie, A. (2008). A generic framework for real-time discrete event simulation (DES) modelling. In *2008 Winter Simulation Conference* (pp. 1931-1938). IEEE.

Zhang, X. (2018). Application of discrete event simulation in health care: a systematic review. *BMC health services research*, *18*(1), 1-11.

Werker, G., Sauré, A., French, J., & Shechter, S. (2009). The use of discrete-event simulation modelling to improve radiation therapy planning processes. *Radiotherapy and Oncology*, *92*(1), 76-82.

Chemweno, P., Thijs, V., Pintelon, L., & Van Horenbeek, A. (2014). Discrete event simulation case study: Diagnostic path for stroke patients in a stroke unit. *Simulation Modelling Practice and Theory*, *48*, 45-57.

Demir, E., Gunal, M. M., & Southern, D. (2017). Demand and capacity modelling for acute services using discrete event simulation. *Health Systems*, *6*(1), 33-40.

Patel, A., & Jernigan, D. B. (2020). Initial public health response and interim clinical guidance for the 2019 novel coronavirus outbreak—United States, December 31, 2019–February 4, 2020. *Morbidity and mortality weekly report*, *69*(5), 140.

Brooks, R., Kotiadis, K., & Van Der Zee, D. J. (2010). Conceptual modeling for discrete-event simulation (Vol. 59). S. Robinson (Ed.). Boca Raton: CRC press.

Peña-Graf, F., Órdenes, J., Wilson, R., & Navarra, A. (2022). Discrete Event Simulation for Machine-Learning Enabled Mine Production Control with Application to Gold Processing. *Metals*, *12*(2), 225.

Creighton, D. C., & Nahavandi, S. (2002). Optimising discrete event simulation models using a reinforcement learning agent. In *Proceedings of the Winter Simulation Conference* (Vol. 2, pp. 1945-1950). IEEE.

Lang, S., Behrendt, F., Lanzerath, N., Reggelin, T., & Müller, M. (2020). Integration of deep reinforcement learning and discrete-event simulation for real-time scheduling of a flexible job shop production. In *2020 Winter Simulation Conference (WSC)* (pp. 3057-3068). IEEE.

Cortes, E., Rabelo, L., Sarmiento, A. T., & Gutierrez, E. (2020). Design of Distributed Discrete-Event Simulation Systems Using Deep Belief Networks. *Information*, *11*(10), 467.

Chang, W. J., & Chang, Y. H. (2018). Design of a patient-centered appointment scheduling with artificial neural network and discrete event simulation. Journal of Service Science and Management, 11(01), 71.

Jackson, I., Tolujevs, J., & Reggelin, T. (2018). The combination of discrete-event simulation and genetic algorithm for solving the stochastic multi-product inventory optimization problem. *Transport and Telecommunication Journal*, *19*(3), 233-243.

Vélez Gallego, M. C., Valencia Ramírez, D. A., & Castro Zuluaga, C. A. (2012). Heurístico de simulación-optimización para la configuración de un sistema de estantería selectiva simple. *Ingeniare. Revista chilena de ingeniería*, *20*(1), 17-24.

Leemis, L. M., & Park, S. K. (2006). *Discrete-event simulation: A first course*. Upper Saddle River: Pearson Prentice Hall.

Dagkakis, G., & Heavey, C. (2016). A review of open source discrete event simulation software for operations research. *Journal of Simulation*, *10*(3), 193-206.

Kilgore, R. A., Healy, K. J., & Kleindorfer, G. B. (1998). The future of Java-based simulation. In *1998 Winter Simulation Conference. Proceedings (Cat. No. 98CH36274)* (Vol. 2, pp. 1707-1712). IEEE.

Garrido, J. M. (2012). *Object-oriented discrete-event simulation with java: A practical introduction*. Springer Science & Business Media.

Section 5

Robinson, S. (2015). Modelling without queues: adapting discrete-event simulation for service operations. *Journal of Simulation*, *9*(3), 195-205.

Perales, W. (2001). Classificações dos sistemas de produção. Encontro nacional de engenharia de produção. Anais.

Section 7

Rossetti, M. D. (2008). Java Simulation Library (JSL): an open-source object-oriented library for discrete-event simulation in Java. International Journal of Simulation and Process Modelling, 4(1), 69-87.

Appendix C

Selfridge, R. G. (1955). Coding a general-purpose digital computer to operate as a differential analyzer. In *Proceedings of the March 1-3, 1955, western joint computer conference* (pp. 82-84).

Fishman, G. S., & Kiviat, P. J. (1968). The statistics of discrete-event simulation. *Simulation*, *10*(4), 185-195.

Kleinrock, L. (1970). Analytic and simulation methods in computer network design. In *Proceedings of the May 5-7, 1970, spring joint computer conference* (pp. 569-579).

Hixson, H. G. (1969). Equipment maintenance studies using a combination of discrete event and continuous system simulation. *Institute of Electrical and Electronics Engineers (IEEE).*

Hillman, L. W. (1969). An order picking and shipping model. *Institute of Electrical and Electronics Engineers (IEEE).*

Wigan, M. R. (1970). Applications of simulation to traffic problems. *Simulation*, *14*(3), 135-136.

Di Febbraro, A., Recagno, V., & Sacone, S. (1970). Intermodal Transportation In Urban Framework: A Discrete-events Simulative Approach. *WIT Transactions on The Built Environment*, *6*.

Jacobson, S. H., Hall, S. N., & Swisher, J. R. (2006). Discrete-event simulation of health care systems. In *Patient flow: Reducing delay in healthcare delivery* (pp. 211-252). Springer, Boston, MA.

11 - APPENDIX

A - Problems & Opportunities

Company wide

- High employee turnover
- Precision in data delivery is essential, slightest errors or misalignment is fatal
- Many cases where there is revenue but no profit
- Combination high customizability & fast turnover is hard to maintain
- Feasibility and coverage checks often missing when bringing in new clients
- Hiring policy for lower tier/skill jobs/tasks is a bottleneck (can take 6 weeks after someone left)

Operations

- Solving database setup problems can take up to 2 months before solved
- Coding - need to adapt to increased demand with extra (Excel) tools which decreases productivity
- Adding retailers done by operations causes serious disruptions in normal workflow
- New retailers causes storm of unknown barcodes
- Sorting EANs for Coding based on revenue share of total market is not sufficient for retailer clients with different inventory, leads to oversight and client unhappiness
- Updating databases happens every 4 weeks (2nd after 1st), can cause serious delay in answering clients requests
- Requests for modifying, updating or other can pile up and take a while before picked up
- Coding - EAN changes by client get rarely identified, which results in misalignment of data analysis
- Often rework necessary
- Measurement Science - Very high variability of requests
- S&R - problems with new system speed, which forces using old system
- S&R - tools and/or knowledge missing to handle returned database files from client
- DBA- Quality is a problem
- Data input is the source of operations, problems there have strong effects on the rest of the company
- Data Input - retailers do not always deliver data consistently
- Data input - a lot of manual labour

Technology & Innovation

- Computer is sometimes bottleneck with handling large datasets
- Innovation suffers from decreased amount of interaction with colleagues
- Separate department, not always clear who is responsible for what in rest of company
- Communication and alignment with international teams can be difficult (India)

Commercial

- Things that are promised to are not always possible with current operational resources
- Not always clear who can handle client requests
- Things sold by commercial not efficiently picked up by operations
- No quantitative analysis client requests -> possibly increase productivity doing more promising requests first
- Pipeline jobs overview (multiple) excel files
- Many people are new in the department, affecting knowledge, providing answers
- Communication about client happiness or trends is often lacking, which limits proactive working possibilities

Opportunities

- Application of forecasting techniques on client trends and moves to allow early adaptation and limits company having to chase
- Research into quantitative request importance analysis
- Research into what is needed to increase company operations flexibility
- New data integration is key for IRI
- Single task training is quick and possible
- DBA - differentiating turnover time delivery & Placement
- Database integration, what opportunities does it provide and what is necessary?
- Achieving cost efficiency using a subscription based “spotify” model
- Do all activities really require a high level of expertise?
- Allowing high customization will increase variability and provide higher risks and higher strain on the operational process
- Orchestrate data handovers in process
- Ability to take decisions more quickly
- Working and delivering in steps with new customers/suppliers, not immediately EAN level
- More organisation around the customer, ordered and clear information is often lacking

B - Search Terms

The following search terms have been used to search for suitable articles in the Scopus and Google Scholar databases:

- Manufacturing vs Services industry
- Production processes under uncertainty
- Queueing networks/models
- Stochastic Programming
- Simulation
- Discrete Event Simulation
- Hybrid Discrete Event Simulation
- Discrete Event Simulation Healthcare
- Discrete Event Simulation Service Industry
- Open Source Discrete Event Simulation Packages
- Discrete Event Simulation in General Purpose Programming Language
- Discrete Event Simulation in Java

C - Short history of Simulation

Although computer simulation can be dated back to (mid) 1950s with articles like R.G. Selfridge (1955) where digital computer systems are coded for differential analysis, first mentions and applications of DES date back to 1968 - 1970. G.S. Fishman & P. J. Kiviat (1968) introduce the complications that are inherent to stochastic system simulation models. L. Kleinrock (1970) provides possible solutions to these statistical problems that are directly related to simulation systems with stochastic behaviour where events occur at a countable set of points in time. Operations Research related applications include H.G. Hixson (1969) with his use of DES (and also continuous simulation) to simulate age-specific failure rates within populations of equipment to determine equipment maintenance policies. Also L.W. Hillman (1969) makes use of DES to design an order picking and shipping system. Other applications of simulation and DES in the same time period include traffic and transportation networks and the development of several simulation languages. M.R. Wigan (1970) summarises preceding research in this domain dating back to 1954. A. Di Febbraro et al. (1970) describe the behaviour of an intermodal integrated urban transportation network with the use of a DES model. S. Robinson (2005) describes the period from 1950 to the 60s as the pioneering period where the foundations for simulation were laid with the use of the first computers and machine code. Followed was the period of innovation, with the development of simulation languages and early animation and interactivity. With the introduction of microcomputers, Robinson (2005) considers the 1980s as the revolution of simulation with the increasing ease of development, wider availability of simulation software and wider use and adoption of simulation in general.

D - Pseudocode of key elements of the Kotlin implementation

Event Actions

Definition of actions to perform on planned events, mainly to handle the start and end of days on which certain tasks are performed. For the general example we assume the entity is an EAN, however sometimes in the simulation it will be the container entity STUB. Two general examples are given by the following pseudocode:

```
Class EventAction StartWorkdayProcessStage()
      Function action(event)
            Check station combine rule
            While (workload queue corresponding to station IS NOT EMPTY)
                  ean = Queue.RemoveNextEntity()
                  nextStation = workloadDivisionRule(ean)
                  nextStation.receive(ean)


Class EventAction EndWorkdayProcessStage()
      Function action(event)
            Check if every workload queue of stage is empty
            Yes →
                  For (station in stations)
                        ean = workloadqueue.RemoveNextEntity()
                        stationsNextStage[station].receive(ean)

            No →
                  For (station in stations)
                        While (stations[station].workloadqueue.isNotEmpty)
                              Ean = workloadqueue.removeNextEntity()
                              ean.daysNotFinished += 1
                              If (ean.daysNotFinished > lateRule)
                                    Ean.late = true
                                    Increase ean priority
                                    reworkQueue.enqueue(ean)
                              Else
                                    reworkQueue.enqueue(ean)
```

ReceiveQObjectInterfaces

Definition of what happens to an entity after it completes processing at its corresponding station. For the general example we assume the entity is an EAN, however sometimes in the simulation it will be the container entity STUB.

```
Class reiceiveQObjectInterface AfterProcessingStage()
      Function receive(entity)
            ean = entity
            Update counters
            Check if ean has late modifier
            Yes →
                  Ean.late = false
```

```
        Decrease ean priority
    If (ean.reworkEvent)
        // for a rework event a single action needs to be performed
        Ean goes directly to the dispose/drain function
    Else
        // a rework status is set if an action is not completed with
        success
        If (ean.reworkstatus is equal to 0)
            If (action is completed with success is true)
                ean.completionState = stagenumber
                nextQueue.enqueue(ean)
            Else
                // action not completed with success
                ean.reworkStatus = stagenumber
                ean.completionState = stagenumber
                // whether an action is successfull get discovered
                at a next stage, not immediately
                nextQueue.enqueue(ean)
        Else
            // unsuccessful action has been discovered and ean has
            been sent back in the process and is completed again
            ean.reworkStatus = 0
            ean.completionState = stagenumber
            nextQueue.enqueue(ean)
```

D - Parameter values of the baseline process

```
private val seconds_per_workday: Int = 28800 // workday of 8 hours
private val productionPeriodLength = 20 //days
private val numberOfKeycatEmployees = 0
private val numberOfKeycatEmployeesTraining = 0
private val numberOfCodingEmployees = 13
private var numberOfCodingEmployeesTraining = 0
private val numberOfPlacementEmployees = 5
private var numberOfPlacementEmployeesTraining = 0
private val numberOfSupportEmployees = 3
private var numberOfSupportEmployeesTraining = 0
private val productivity = 0.8
private val upcArrivalMean = 2888
private val upcArrivalSdv = 740
private val autoEnginePercentage = 0.3
private val importantKeycatPercentage = 0.4
private val proactiveKeycatPercentage = 1 - importantKeycatPercentage
private val firstTimeRightKeycat = 0.95
```

```
private val firstTimeRightCoding = 0.95
private val firstTimeRightPlacement = 0.95
private val percentageNilWithFeedback = 0.02
private val percentageOnlyKeycatAssignment = 0.40
private val keycatLateRule = 2  //days
private val codingLateRule = 2 //days
private val placementLateRule = 4 //days
private val eligibleForKeycat = 0.9
private val keycatSingleQueue = false
private val codingSingleQueue = false
private val placementSingleQueue = false
private val keycatCodingTaskTogether = false
private val keycatCodingSingleQueue = false
private val keycatWorkloadDivisionRule = 1
private val codingWorkloadDivisionRule = 1
private val placementWorkloadDivisionRule = 1
```