Universiteit Leiden
The Netherlands

Opleiding Informatica & Economie

Real-time Assignment of Service Engineers based on Multiple Criteria Optimisation

C.J. Klein Essink

s1818279

Supervisors:
M.T.M. Emmerich (LIACS)
Y. Wang (LIACS)
R. Kroonenberg (Siemens, NL)

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl 16/08/2019

Abstract

Dispatchers, of the Building Technologies division at Siemens NL, are still manually assigning service engineers to service calls. Sending the right service engineer to each defect or problem is imperative to customer satisfaction, by means of the rate of successful service at first attempt or visit (First Time Right). So how do dispatcher choose? Which criteria are weighed consciously and subconsciously? This paper focuses on building a model that recommends per situation, in real-time, the best service engineer to send based on multiple criteria optimisation; a model using a modified desirability function approach is implemented and the recommendations visualised on a dashboard, for ease-of-use of the dispatchers.

Contents

Chapter 1

Introduction

At Siemens NL, the business unit Customer Services (Building Technologies division) handles customer calls for corrective maintenance on Siemens products (e.g. systems, hardware, software, etc.). Customer calls first reach the Customer Service Center (CSC) via e-mail, telephone, or the service portal after which operating dispatchers at Customer Services follow up and send the right service engineer to perform corrective maintenance. Sometimes they send service engineers who are close by, in case of an emergency, or the most experienced veteran in highly complex situations.

Currently on average, 56% of the performed maintenance is First Time Right. First Time Right is a KPI (Key Performance Indicator) that calculates the rate of successful service at the first attempt/visit. Siemens aims to achieve higher customer satisfaction, thus wants to maximise the First Time Right metric. To do this, firstly, the right service engineer should be sent out on every service call. Optimising the process of choosing the right service engineer to handle each service call, is the focus of this paper.

Briefly speaking, the goal is to develop a model that takes the set of available service engineers and outputs the optimal assignment of engineers to unassigned service calls. This formed the basis for the following research question.

> *How to best implement and design a model for real-time assignment of service engineers based on multiple criteria optimisation?*

Since the model is intended as an aid to dispatchers, model output (best assignment recommendations) is to be presented on a dashboard. Before the project started, a mock dashboard was created, as a possible end result (See Appendix A, Figure A.1).

Thesis Overview

Choosing the right service engineer can be done using many criteria also called objectives; optimising such situations with multiple, possibly contradictory objectives is called multiobjective optimisation (MOO) also called multicriteria optimisation (MCO), which is an area of multicriteria decision analysis (MCDA) concerning mathematical optimisation of not one but multiple objective functions. The discipline of MCDA supports decision makers in situations involving uncertainty regarding optimal solutions (e.g. planning, choosing between product/material alternatives, etc.) using the decision makers preference, explicitly or otherwise implicitly stated.

Theories from the field of MCDA will provide the theoretical background for this thesis. First, the basic concepts will be introduced in Chapter 2. These concepts provide the foundation for the related work detailed in Chapter 3. Chapter 4 aims to unite abstract theory with the applied setting at Siemens. Chapter 5 then describes the methodology used in the implementation, found in the next chapter, Chapter 6. Following concluding statements on this thesis, set forth in Chapter 7, some recommendations on furthering development and research are discussed in Chapter 8.

I would like to thank Michael and Robin first and foremost for their time, supervision, and invaluable insights. Thank you Yali, for your help and feedback! Lastly, thank you Mike for your assistance. I am grateful to have been given the opportunity to write my bachelor thesis at Siemens NL and LIACS.

Chapter 2

Basic concepts

This chapter aims to explain the general approach of solving multiobjective optimisation problems. This approach forms the basis of the structured approach used in this thesis, detailed in the methodology chapter, Chapter 5.

Decision space

First, let X be the set of all possible decisions at a given moment, called decision space. Every item in X is a real object, item, material, product or other entity which is to be the subject of MCDA. For example, in this project the set of all available service engineers. Every single option (here: a service engineer) in this space can be represented by a decision vector $\vec{x} \in X$ [JRFE05].

Objective space

Secondly, to evaluate for every identified decision alternative $\vec{x} \in X$, an objective function $\vec{f}$, to the objective space $\mathbb{R}^m$, is used where m is equal to the number of objectives/criteria/attributes (from now on objectives) in the objective space. Objective functions can be written as such: $\vec{f} : X \to \mathbb{R}^m$ or $f_1, f_2, \dots, f_m$ with $f_i : X \to \mathbb{R}$ and $i = 1, 2, \dots, m$. This does not mean that every objective vector $\vec{y} \in \mathbb{R}^m$ necessarily has a preimage in X, i.e. is or can be represented by a decision vector $\vec{x}$. Of $\mathbb{R}^m$ the feasible objective region Z comprises the range of objective vectors, $\vec{y} \in Z$, all objective functions $\vec{f}$ can map to: $Z \subseteq \mathbb{R}^m$ [Mie98]. Figure 2.1 visualises the mapping using objective functions from X to Z.

As dealing with multiple objective functions is complex and is usually avoided, Haftka states that there are two ways that can be used for reducing the number of objective functions. One way is an aggregation approach, that summarises all objective functions in a single aggregated objective function. The second, is to select only one main objective function and limit the others [HG12].

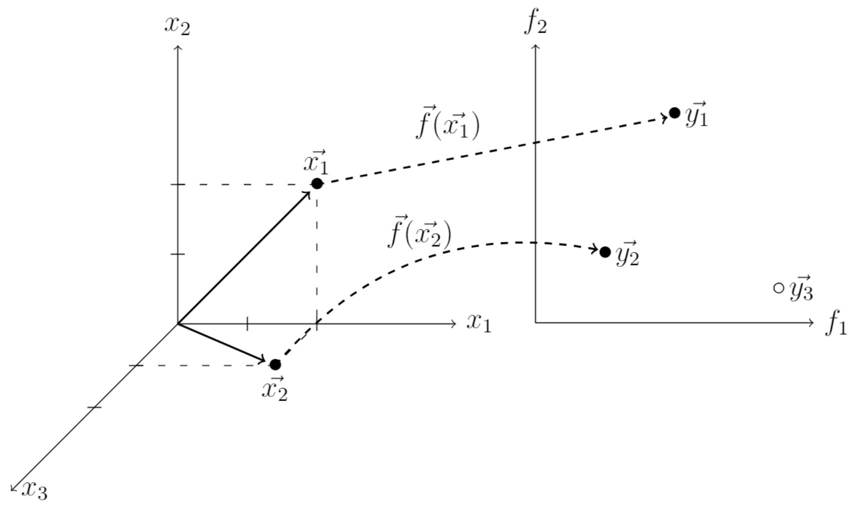


Figure 2.1: Visualisation of $\vec{f} : X \to \mathbb{R}^m$. NOTE: $\vec{y_3}$ has no preimage in X, i.e. $\vec{y_3} \notin Z$.

Ranking and comparing

One must realise that the value ranges of objective functions can differ: they may also be incommensurable (i.e. in different units) [Mie98]. In many cases it is advantageous to transform the original objective functions; in most applications the most robust approach is normalisation [MA04]. So, if necessary, normalisation, of the objective function or the values of all objective vectors $\vec{y}$, after mapping from $x \to Z$, is applied.

Viewing a decision as an objective vector with all its attributes $x_1, \dots, x_m$ is convenient when comparing alternatives in MCDA. To illustrate this point, suppose one has two objective vectors (decision alternatives), as shown in example 1, and information that all objectives are subject to maximisation. With this information, ranking these alternatives can be done manually.

$$\vec{V_1} \begin{pmatrix} 9 \\ 8 \\ 2 \end{pmatrix}, \vec{V_2} \begin{pmatrix} 4 \\ 7 \\ 4 \end{pmatrix}$$

Example 1: two objective vectors comparison

Vector $\vec{V_1}$ scores higher on attributes one and two, and slightly lower on the third attribute. Intuitively, without preferences indicating that attribute three is extremely important, vector $\vec{V_1}$ would be the better choice.

In MCDA, all feasible alternatives are not always explicitly known in advance. A possibly infinite number of them exists and one has to generate the alternatives before they can be evaluated. To evaluate every decision alternative $\vec{x} \in X$, the alternatives have to be ranked. The problem of ranking could be reduced to finding the 'maximum' in a given set. This is easily possible if one alternative clearly dominates every other alternative. The concept of domination has a formal definition.

Given two objective vectors $\vec{y}_{1,2} \in \mathbb{R}^m$ described by the same properties $Y_i(\vec{y})$, $i = 1, \dots, m$, it is

> said, that $\vec{y}_1$ dominates $\vec{y}_2$ ($\vec{y}_1 >> \vec{y}_2$), if $Y_{i1} \geq Y_{i2}\ \forall i = 1, 2, \ldots, m$ and $Y_{j1} > Y_{j2}$ for at least one $1 \leq j \leq m$.

Domination has an important drawback, because there may not be a single best solution dominating all others. And worse, as $|Z|$ increases, it becomes more and more unlikely for an object to dominate any other [Mie98]. Moreover, when dealing with a large number of objective functions it is not often the case that one solution dominates the others. Example 1 shows that even with only two vectors described by three attributes, it is not probable that one vector completely dominates the other(s). Vector 1 is practically better but does not meet the strict definition of domination ($1 : x_3 \ngeq 2 : x_3$).

To deal with this, a less strict concept of Pareto-optimality can be used. It is defined as follows [Mie98]:

> Given a set $X \subset \mathbb{R}^m$ of objects, an object $P \in X$ is called Pareto-optimum, if there is no object $Q \in X$ with $Q >> P$

Pareto-optimality can be defined as a property for a solution to make one objective function value better would imply that another objective function value gets worse. On the other hand, at a non-Pareto-optimal point, the solution can still be improved in at least objective, without getting worse in any other objectives.

Scalarisation

Evaluating for higher-dimensional spaces (i.e. an objective space with $m > 3$) is more difficult and complex than for two or three dimensions; as stated, the chances of domination dwindle and there are more Pareto-optimal solutions. Therefore, thirdly, for evaluating decision alternatives, a utility function is used to map from objective space to a single scalar. This is called scalarisation, which means converting a problem into a single or a family of single objective optimization problems with a real-valued objective function, termed the scalarising function, depending possibly on some parameters [Mie98]. A scalarising utility function is formally written as such: $U : \mathbb{R}^m \to \mathbb{R}$. To illustrate the simplicity, consider the following two-objective scenario, using weighted sum approach [JRFE05]:

$$U(\vec{x}) = 0.3 \times \vec{f}_1(\vec{x}) + 0.7 \times \vec{f}_2(\vec{x})$$

Example 2: a scalarising utility function

There are many types of scalarising utility functions; most revolve around products or sums of the objective functions, including weights and means.

Evaluation

Lastly, ranking can be done by sorting on the basis of either minimising or maximising the real-valued output of the utility function, for every alternative in the Pareto-optimal set.

	Maximising		Minimising	
Rank	Service engineer	Value: U(x)	Service engineer	Value: U(x)
1	Mr. 12	9.6	Mr. 45	3.3
2	Mr. 67	7.9	Mr. 34	6.2
3	Mr. 89	7.7	Mr. 89	7.7
4	Mr. 34	6.2	Mr. 67	7.9
5	Mr. 45	3.3	Mr. 12	9.6

Example 3: ranking based on maximising and minimising

Chapter 3

Preliminaries & related Work

This chapter will focus on MCO approaches and methods, which can be classified in many ways according to different criteria. In 1979 Hwang and Masud presented a classification of methods according to the participation of the decision maker in the solution process [Mie98]. The classes are:

1. No-preference methods: preference information is not utilised.
2. A posteriori methods: articulation of preferences is used after optimisation. A typical a posteriori method is to compute a Pareto-optimal set.
3. A priori methods: the optimal solution is found by the search method after the DM has already stated all preferences.
4. Interactive methods: is an iterative process where in each iteration the DM can change or add preference information.

Overlapping and combinations of classes are possible and some methods can be considered to belong to more than one class. In the context of the research question, the 'a priori' approach is of interest, further detailed in the following section. No-preference, interactive, and a posteriori methods are not relevant, thus will not be featured.

3.1 A priori

In the case of a priori methods, the DM must specify his or her preferences, hopes and opinions before the solution process; this is often difficult, as a DM does not necessarily know beforehand what is possible to attain and how realistic his or her expectations are. A priori articulation of preference information by the DM starts with designing a function and then finding a single optimum with respect to the function. This means that the DM is involved before optimisation [Mie98].

A common approach to imposing constraints is to develop an aggregate or utility function [MA04]. The second a priori method featured in this overview is lexicographic ordering and lastly, goal programming is introduced.

3.1.1 Utility functions

Within MCO, utility/value functions are classified under Multiattribute Utility Theory (MAUT). The purpose of MAUT is to represent the preferences of the DM on a set of alternatives, X, by an overall utility function [GEF10]. As $U : \mathbb{R}^m \to \mathbb{R}$, the utility function problem

maximise or minimise

$$U(\vec{f}(\vec{x}))$$

subject to

$$\vec{x} \in X$$

can be solved by some method for single objective optimisation. This function computes a value for every decision alternative and allows for every pair to be ordered, based on this value [Mie98].

Utility functions have two main classes; additive and multiplicative forms. If $m \geq 3$, and for some $x_i \in X$, then

$$U(\vec{x}) = \sum_{i=1}^{m} w_i \times U_i(\vec{x}_i), \tag{3.1}$$

or

$$1 + w \times U(\vec{x}) = \prod_{i=1}^{m} \left[1 + w \times w_i \times U_i(\vec{x}_i)\right], \tag{3.2}$$

where U and the U_i are utility functions scaled from zero to one, the w_i are scaling constants with $0 < w_i < 1$, and $w > -1$ is a nonzero scaling constant [Kee74].

If a DM's preference structure is additive, for example, for two objectives x_1 and x_2 and given $U(x_1, x_2) = U_1(\vec{x}) + U_2(\vec{x})$, the utility function is of additive form as well [KRWR79]. Consequently, for multiplicative preferences, the utility functions form is multiplicative.

Desirability functions

Theoretically still utility functions, but a special mention should be made for desirability functions. Desirability functions were first introduced by Edwin C. Harrington Jr. in 1965 [Har65]. Their aim is to factor in what other MCO methods try to ignore, i.e. incommensurability of targets. In the formal description of MCO problems, we find m different targets, from $x_1, x_2, \ldots, x_m \in X$, to be optimised. Translated to real world problems it means that up to m different scales of measurement (e.g. times, lengths, weights, etc.), have to be transformed to be commensurable [Ste99].

The transformation is done by designing a specific desirability function for each objective mapping to $[0, 1]$. Any function mapping from domain $\mathbb{R} \to [0, 1]$ can be called a desirability function d, under the following conditions [Ste99]:

1. d should be flexible and accommodate for different problem formulations
2. It approaches zero as the objective function value goes to $-\infty$ or ∞. The highest value is obtained at the target value T. It decreases to zero monotonically

After scaling all targets individually to a desirability scale, they have to be combined to a single number, the overall desirability index of a decision alternative can be used to evaluate. A desirability index is a function D with D: $[0, 1]^m \to [0, 1]$. Notice that desirability functions are utility functions $U : \mathbb{R}^m \to \mathbb{R}$. Harrington used the following exponentials for target value problems: $d^H(Y) := e^{-|Y'|^m}$, where Y' is an appropriate transformation of Y. Appropriate in the sense of Harrington is to choose Y' in a way that $d^H(LSL) = d^H(USL) = 1/e$. As a possible transformation he gives $Y' = (2Y - (USL + LSL))/(USL - LSL)$, where LSL is the lower specification limit and USL defines the upper specification limit [MT06]. Harrington-type desirability functions are designed to be symmetric around the centre between the LSL and USL.

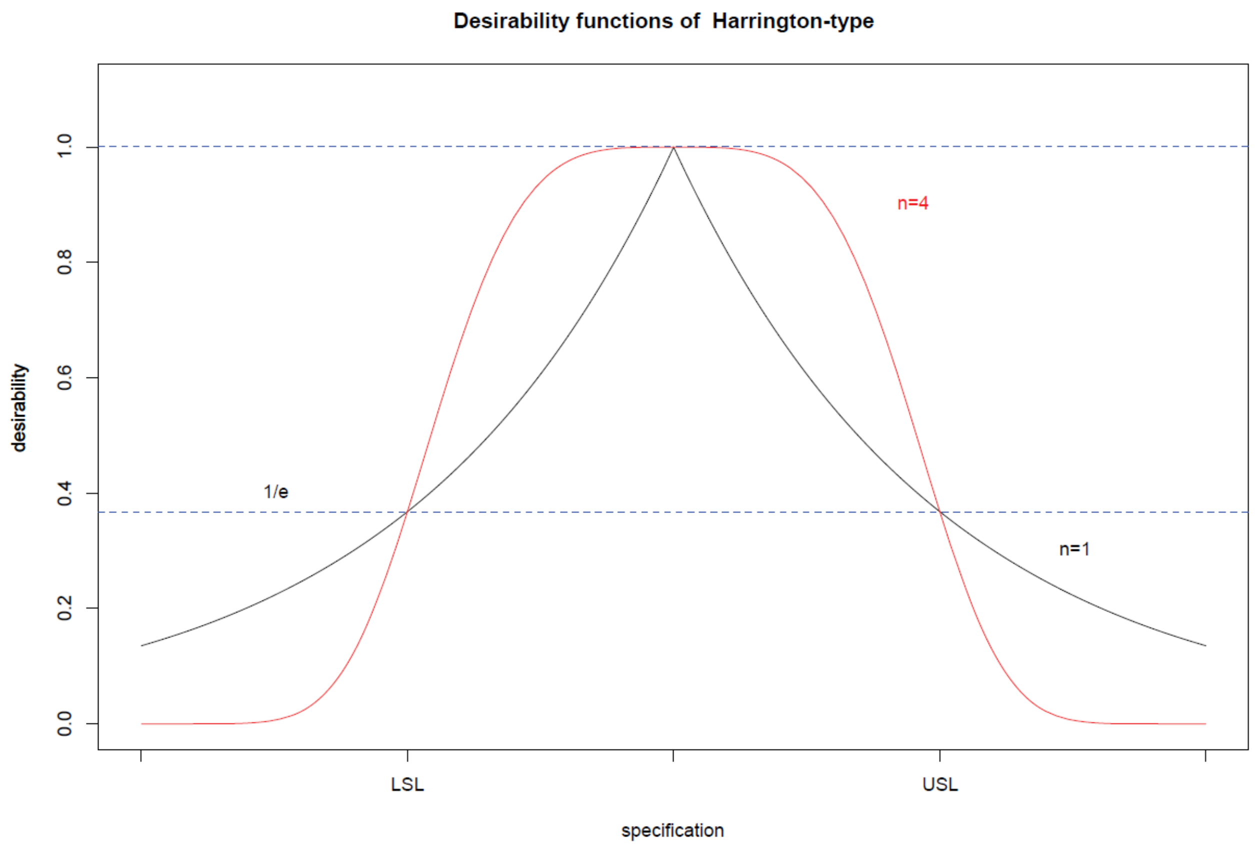


Figure 3.1: Desirability functions of Harrington-type for two different values of n(m) [Ste99]

Interpretation of Harrington's desirability index is done using desirability function value ranges. This effectively provides a grading scale for ranking (the items below are stated exactly as in the original reference [Har65]):

— desirability 1: ″ultimate satisfaction″ or ″improvement beyond this point has no value″

— desirability 0.8 - 1: ″excellent″ or ″well beyond anything available″

— desirability 0.63 - 0.8: ″good″ or ″slight improvement over industrial quality″

— desirability 0.4 - 0.63: ″acceptable, but poor″

— desirability 0.3 - 0.4: ″borderline″

— desirability 0 - 0.3: ″unacceptable to completely unacceptable″

In literature about applications of desirability functions the most common forms of desirability functions are those of Harrington and Derringer-Suich [Ste99]. These desirability functions are also two-sided, but additionally feature one-sided transformations [DS80].

$$\textit{Two-sided } d_i = \begin{cases} 0 & \text{if } f_i < LSL_i \\ \left(\frac{f_i - LSL_i}{T_i - LSL_i}\right)^{s_i} & \text{if } LSL_i \le f_i \le T_i \\ \left(\frac{f_i - USL_i}{T_i - USL_i}\right)^{s_i} & \text{if } LSL_i \le f_i \le USL_i \\ 0 & \text{if } f_i > USL_i \end{cases}$$

One-sided Larger-The-Best (LTB) transformations [CLP11] are desirability functions with beneficial objectives subject to maximisation [KGC13]:

$$\textit{One-sided beneficial } d_i = \begin{cases} 0 & \text{if } f_i < LSL_i \\ \left(\frac{f_i - LSL_i}{T_i - LSL_i}\right)^{s_i} & \text{if } LSL_i \le f_i \le T_i \\ 1 & \text{if } f_i \ge T_i \end{cases}$$

For one-sided Smaller-The-Best (STB) transformations, for non-beneficial objectives, the desirability function be rewritten as below:

$$\textit{One-sided beneficial } d_i = \begin{cases} 1 & \text{if } f_i < Ti \\ \left(\frac{f_i - USL_i}{T_i - USL_i}\right)^{s_i} & \text{if } LSL_i \le f_i \le USL_i \\ 0 & \text{if } f_i \ge USL_i \end{cases}$$

T_i is a target value representing the highest value for a criterion to be maximised (beneficial) and represents the lowest value for criteria to be minimised (non-beneficial), unless other specific target value is defined [KGC13]. The shape of the curve, defined by s, has implicit risk preference information; showing whether the DM is risk averse, risk prone, or risk neutral. Thus, risk preference is applied by choosing a shape of the desirability transformation and value for s [KM04]. Visualised in Figure 3.2, are two examples of one-sided transformations.

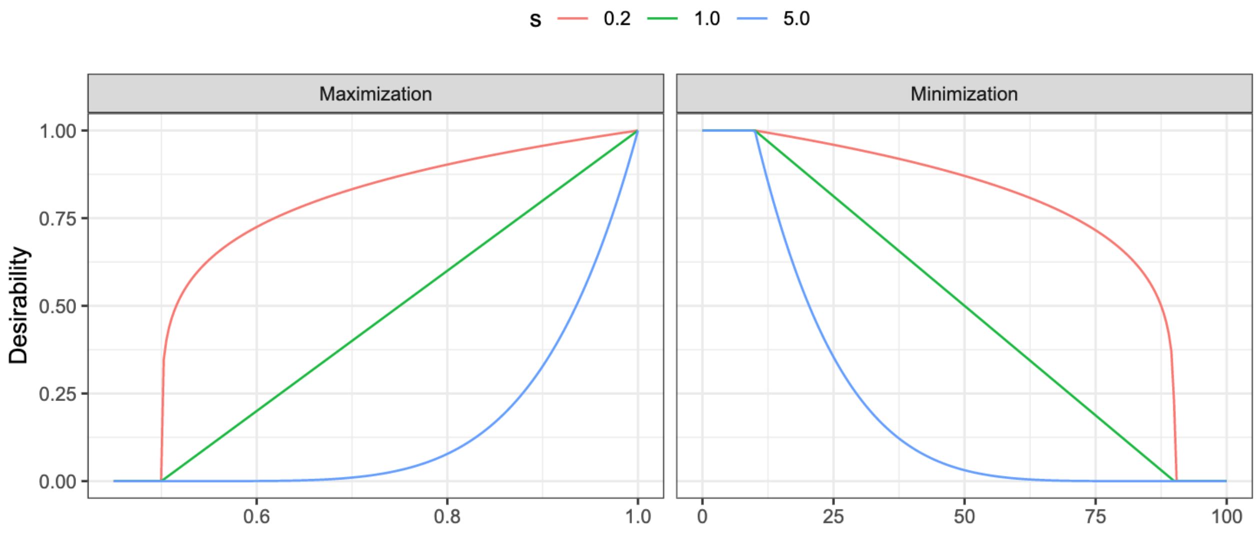


Figure 3.2: One-sided desirability functions. Left: LTB transformation. Right: STB transformation [KJ19]

3.1.2 Lexicographic ordering

In lexicographic ordering the objectives are sorted by priority and first it is checked if the highest priority objective gives a difference in the objective function value. If there is a difference one would choose based on the highest priority objective function. If not, the second highest priority objective function is chosen. This process goes on as above [JRFE05].

3.1.3 Goal programming

Goal programming is on of the first methods expressly created for MCO. The basic idea in goal programming is that the DM specifies (optimistic) aspiration levels for each objective function f_i and any deviations d_i from these aspiration levels are minimised. An objective function combined with an aspiration level forms a goal, where goals and constraints are effectively of the same form. This is why the constraints may be regarded as a subset of the goals [JRFE05].

Then, the total deviation from the goals of all objective functions $\sum_{i=0}^{m} |d_i|$ is minimised by choosing the decision alternative $\vec{x}$ with the lowest overall deviation [MA04].

3.2 Summary

Multiple a priori methods are introduced above. Each method approaches ranking in a different way and uses specific value types and output ranges. In this context, desirability functions are the most suitable method

Chapter 4

Evaluation

In order to determine the best approach for designing and implementing a model for real-time assignment of service engineers, a thorough assessment/analysis of the system context is necessary. Combining the theoretical framework in Chapter 3 with the following contextual analysis, some key considerations follow, which influenced design choices in methodology 5.1 and implementation in Chapter 6.

4.1 Context

Management of Customer Services, within Building Technologies (BT) division, Siemens NL, has divided the map of the Netherlands into three geographical regions named Y1, Y2, and Y3. Service engineers are currently assigned tasks based on their respective working region, in turn determined by their home address. By assigning work, exclusively within the bounds of 'their' region, Siemens aims to cut down on average traveling distance to customers. Thus reducing travel time (time not spent 'working' and effectively the response time), fuel related and other car related costs, and the collective carbon footprint (One of Siemens' global sustainability strategy goals is to operate carbon neutrally by 2030 [Sie19a]).

Siemens NL employs a large variable group of service engineers, who are primarily tasked with maintenance for Customer Services (BT). There are two types of maintenance service calls: for preventive maintenance and corrective maintenance. Service calls come in as notifications for the dispatchers at BT, they then have to assign the notification to a service engineer. All service engineers employed by Siemens NL are trained and certified to work on virtually all of Siemens customer installations. Although, maintenance practices for certain (very) old installations are not part of the standard training anymore. Meaning that those old installations, a type within Siemens classified as "old techniques", can only be serviced by a select group of service engineers that have many years of experience.

Preventive maintenance

Preventive maintenance is pre-planned work, previously planned and scheduled manually by schedulers, but, currently assigned to engineers based on multiple criteria by an automated scheduler. A KNIME workflow is used to implement this automated scheduler. KNIME is an open-source data analytics platform, well suited for handling multiple data sources [AG19]. The frequency and extent of maintenance is determined by the service level agreement between Siemens and each customer. The workflow also takes into account holidays, weekends, stand-by days, contractual working hours, all working schedules of available service engineers, and lastly it minds per task, to be scheduled, which service engineers are qualified to perform maintenance on all particular installation types. When scheduling, each day a sub-group of ten service engineers are kept 'free' for corrective maintenance, when assigning tasks. All schedules and relevant information are saved in Siemens ERP systems and local databases (e.g. SAP and Microsoft SQL servers). Even though scheduling is now automated, Customer Services (BT) still employs dispatchers for preventive maintenance, tasked with customer services, service engineer management, and rescheduling due to unforeseen circumstances. Dispatchers utilise SAP for oversight on daily/weekly schedules, customer contracts, and customer relations (e.g. contact history, service history, etc.). In addition to SAP a Qlik Sense dashboard was built, providing an overview and visualisation of the KNIME workflow output.

Corrective maintenance

Corrective maintenance is similar to preventive maintenance: both involve the same customers, installations, and the same pool of service engineers that perform maintenance. However, customer service contracts significantly change the role of dispatchers for corrective maintenance service calls: contracts may contain a clause stating an agreed time span within which Customer Services have to respond (effectively meaning to start communication) and an on-site due time, which can even differ between specific defects: on-site times generally are four, eight, or twenty-four hours, whereas defects involving electrical power supplies and accumulators usually require on-site assistance within twelve hours.

Due to contractual obligations and the nature of the service calls, dispatchers for corrective maintenance have to send service engineers with more urgency and immediacy. Currently, dispatchers for corrective maintenance use SAP and maintain a physical list of available service engineers (again, ten service engineers each day). Using this list and the schedules viewed using SAP, service calls are manually assigned to one of the six service engineers on-call, two service engineers per region, and just for cases of emergency, four service engineers are scheduled stand-by (meaning, within Siemens, having free time at home, but remaining available as backup).

4.2 Key considerations

With regards to the scope of this thesis: an efficient system is already implemented for preventive maintenance, moreover, scheduling is not real-time assignment, thus, corrective maintenance will be the focus of the model. Importantly, the scheduler can assign tasks autonomously whereas the model will not assign but merely score as to provide a recommendatory list of service engineers to assign that notification. Since corrective maintenance is assigned to a small group of at most ten service engineers, combinatorial optimisation is not necessary. The possibility of conflicts or suboptimal assignments of service engineers to service calls is low in pairing these small sets (at most ten service engineers to, on average, few service calls at the same time). Additionally, due to the small set of decision alternatives, only calculating based on the Pareto-optimal set is not saving a lot of time and will not be part of the methodology: it might exclude too many decision alternatives.

A KNIME workflow is used for preventive maintenance scheduling; implementing the model by developing a second KNIME workflow is expected to aid maintainability from Siemens' congruent expertise, transparency of the data transformation and overall process inherent to KNIME workflows, and incur zero additional system costs by sharing space on an existing KNIME server (no costs involved with alternative software/server package licensing). For development of the model in KNIME, a priori elicitation of the DM's preferences could prove a good starting point; the shape of both individual objective functions and relationship between the objectives are unknown. Preferences have to be elicitated as prior projects at Siemens NL have not featured a similar objective and explicit preference articulation is inherently impossible. Determining function weights or ratios could, reflectively, provide alternative insights into the nature of the objective functions. Preferences will be elicited from the DM, from hereon collectively representing all dispatchers and involved management within Customer Services (BT), using an interview and questionnaire pertaining objective importance and objective function bounds. After preference elicitation, a modelling approach has to be chosen.

Management of Siemens NL has decided to primarily use Tableau software for visualisation and dashboarding across all divisions. Building Technologies division however, has opted to use Qlik Sense. Since preventive maintenance has an existing dashboard in Qlik Sense, having two similar dashboards, would help interoperability. In addition, dispatchers from preventive maintenance could more easily substitute for corrective maintenance dispatchers in case of absences/sick leave. In light of this, output of the model would be best featured in Qlik Sense for the dispatchers. Unfortunately, getting permission, from the time it was requested, to access the data, in Qlik Sense, required to build the dashboard, would exceed the alloted time; thus, Tableau software will be used to implement the dashboard.

Chapter 5

Methodology

The methodology used for implementation, in Chapter 6, derives from the following structured approach to the scoring of service engineers, based on basic concepts of designing and using a multiattribute utility function for scoring, Chapter 2.

1. Define decision space and objective space
2. Define objective functions and their value range
3. Define a scoring function for each objective
4. Normalise to a unified grading scale
5. Determine relative importance of each objective
6. Scalarise
7. Evaluate

This approach is further detailed in Section 5.1. The resulting formulae are summarised at the end of this chapter, in Section 5.2.

5.1 Motivation and design choices

Within the scope, design choices were motivated by practical considerations, from Siemens' defect maintenance context outlined in Chapter 4, and the following principles of scoring [KR93] [MT06].

I Score different aspects individually

II Scores should be easy to interpret and normalised

III Scoring function shapes should be simple if a complex shape is not necessary

Step 1: Define decision space and objective space

As stated in Chapter 4, the decision space will be limited to the available service engineers for defect maintenance. In effect, the set X changes every day, generally it has ten unique service engineers: $X = \{ID_1, ID_2, \ldots, ID_9, ID_{10}\}$ where ID is a personal identifier.

The number of individual objectives m, thus equal to the number of objective functions, have to be determined; the number of objectives involved have a direct impact on complexity. Complexity reduces the ease of interpretation and intuitiveness which is in conflict with principle II; thus, lower-dimensionality is preferred in most cases. In this case, a more accessible two-dimensional objective space $\mathbb{R}^2$ (ergo, $m = 2$) was chosen, based on the bipartition of a distance or time-to-customer objective and an experience objective. The bounds of objective space are drawn from the value ranges of the two objective functions, defined in *Step 2*.

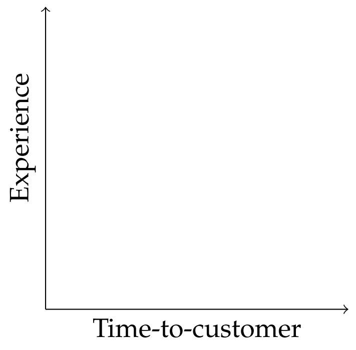


Figure 5.1: Chosen objective space $\mathbb{R}^2$

Step 2: Define objective functions and their value range

Time-to-customer calculation

Every service engineer has a car equipped with a GPS-enabled tracking device from TomTom. In addition to real-time location tracking, WEBFLEET.connect (TomTom API and web-interface) supports route time and distance calculations, for every registered object in a group, to a specified destination. Using this API, time calculations are highly accurate as all route calculations take traffic and diversions into account.

Route times to customer are returned with second accuracy. Intuitively, route times will vary from zero seconds, when a service engineer is already on-site for a preceding defect, to a maximum of arguably five hours to get from the absolute northernmost area in the Netherlands to the southernmost location, or vice versa. Meaning, time-to-customer $\vec{f}_{time}(\vec{x}) \in [0, 18000]$ seconds.

Experience matrix

Experience can be construed as a subjective measure: to say that a service engineer is experienced or not, and categorising how experienced, is subjective. However, using Siemens NL historical data on all service calls of the last thirteen years, experience can be quantified: from this data the type of installation, specific customer installation, and service engineer who performed maintenance, can be extracted. After counting the number of service calls per installation type (in total) and the number of times per specific customer installation, an experience matrix can be constructed to indicate how suited an engineer could be for handling a specific service call. This experience matrix was combined with an existing document that states which installation types each service engineer is allowed to service; this operation added a column 'Certified (0/1)' to be used as a feasibility check before function calculations, assignment, and optimisation.

ID	Installation type	Customer	Certified (0/1)	Service on type (#)	Service for customer (#)
1563	NL5:65347	808906	1	12	3
1563	NL5:65347	808907	1	12	8
1563	NL5:65347	808912	1	12	1

Example 2: obfuscated view of the experience matrix table

Above example shows an excerpt of the experience matrix. The rows show the experience of one service engineer, denoted by the identifier (ID) '1563', regarding installation type 'NL5:65347'. It shows that for this installation type, the service engineer has worked on three different specific installations and performed service a total of twelve times.

Experience, quantified by the experience matrix, is thus mapped to by a hierarchical objective function comprised of two attributes: experience per installation type and specific customer installation, $\vec{f}_{experience} = [\vec{f}_{type}, \vec{f}_{specific}]$. Defining the bounds of $\vec{f}_{experience}$ can be done using the experience matrix and determining the ranges of values for $\vec{f}_{type}$ and $\vec{f}_{specific}$.

At most, currently, a single service engineer has serviced a particular installation type 295 times. The highest number of times a single service engineer has performed on a specific customer installation is equal to 45 times. Therefore, extracted minimal and maximal values from the experience matrix result in $\vec{f}_{type}(\vec{x}) \in [0, 295]$ and $\vec{f}_{specific}(\vec{x}) \in [0, 45]$.

Step 3: Define a scoring function for each objective

The objective functions $\vec{f}_{experience}$ and $\vec{f}_{time}$ both map to non-negative, rounded real values, in $\mathbb{R}$. Thus, commensurability is not an issue in the feasible region Z.

For scoring, the desirability function method is most attractive because it is intuitive, once visualised, and simple, thus, in line with Principle II and III. Desirability also adheres to Principle I, for each objective a specific desirability function should be designed. Moreover, desirability approaches can better balance the optimisation of multiple quality characteristics and provide flexibility in weighing each of them [CLP11]. Lastly, considerating the implementation, desirability functions are suitable for this MCO model as the approach is easily implemented or coded and features a high degree of modularity [KM04].

Desirability functions approach scoring in the following abstract manner: $X \to Z \to [0,1]^m \to [0,1]$. But, Principle II states that scores should be easily interpretable; for the practical application and further development, of this MCO model, at Siemens NL the Dutch grading system will be used for the desirability functions and aggregate, the desirability index. Meaning desirability functions $d_{time}(\vec{f}_{time})$, $d_{experience}(\vec{f}_{experience})$, and $D_{overall}(d_{time}, d_{experience})$ will map to values in the range of zero to ten [Nuf18], summarising: $d_{time} \wedge d_{experience} \wedge D_{overall} \in [0,10]$ by $X \to Z \to [0,10]^m \to [0,10]$. Altering desirability functions to equate to ten-fold the normal output, requires simple modifications.

Modified desirability function forms

Since Siemens NL wants to reduce travel-time and response times, the objective Time-to-customer is subject to minimisation; modifying the one-sided desirability functions of the non-beneficial objective form yields:

$$\textit{Time-to-customer } (d_{time}) = \begin{cases} 10 & \text{if } f_{time} < T_i \\ 10 \times \left(\frac{f_{time} - USL_i}{T_i - USL_i}\right)^s & \text{if } T_i \leq f_{time} \leq USL_i \\ 0 & \text{if } f_{time} > USL_i \end{cases}$$

Scoring for experience is less straightforward; service engineers at Siemens NL are trained and certified before they are employed (Section 4.1), meaning all feasible service engineers are technically equally qualified. During the designing and developing phase, it was deemed unfair to start scoring experience at zero, by fully adhering to the Dutch grading convention. Combining Harrington's desirability index interpretations, see Section 3.1.1, with Dutch grading resulted in the decision to start scoring experience at five, $d_{experience} \in [5,10]$; in the range of 0.4 to 0.63 (in the context of this thesis these values equate to 4.0 and 6.3) interpretation of a desirability index is "acceptable but poor" and sufficient marks are greater or equal to 5.5, within Dutch grading practices.

It is best to send the most experienced service engineer in order to increase the likelihood of First-time-right; rendering $d_{experience}$ subject to maximisation, in contrast with d_{time}. Additionally, experience is a hierarchical objective, the 'specific experience' a service engineer has with the particular customer installation of a notification will be included as a bonus, using $f_{bonus}(\vec{f}_{specific}) \in [0,10]$, as a remainder of a perfect score (i.e. a score equal to ten). This design decision resulted in the following abstract equation for $d_{experience}$:

$$\textit{Modified overall experience}(d_{experience}) = d_{general} + \left((10 - d_{general}) \times \frac{f_{bonus}}{10} \times \alpha\right)$$

Here, α is a variable that determines the weight of bonuses awarded, representing the percentage of the remainder the DM is willing to give as bonus: $\alpha \in [0,1]$, where for example 0.3 awards, potentially, 30 per cent of the remainder if $f_{bonus} = 10$.

Ensuring that overall $d_{experience} \in [5, 10]$, first the general experience desirability score $d_{general} \in [5, 10]$ is calculated. If $d_{general}$ is equal to ten, no bonus needs to be awarded.

$$\textit{General experience } (d_{general}) = \begin{cases} 10 & \text{if } f_{type} \geq T_i \\ 5 + (10-5) \times \left(\frac{f_{type} - LSL_i}{T_i - LSL_i}\right)^s & \text{if } LSL_i \leq f_{type} \leq T_i \\ 5 & \text{if } f_{type} < LSL_i \end{cases}$$

In light of Principle III a simple linear function was chosen for f_{bonus}.

$$\textit{Specific experience}(f_{bonus}) = \frac{10}{\beta} \times f_{specific}$$

Where $\beta \in [1, \infty]$ is a value, specified by preference elicitation, indicating the number of service calls needed to receive a maximum bonus.

Combining and substituting $d_{general}$ in the equation of $d_{experience}$ gives the following overall experience desirability function.

$$\textit{Experience}(d_{experience}) = \begin{cases} 10 & \text{if } f_{type} \geq T_i \\ \text{General score: } 5 + (10-5) \times \left(\frac{f_{type} - LSL_i}{T_i - LSL_i}\right)^s & \text{if } LSL_i \leq f_{type} \leq T_i \\ \text{Bonus: } \left(10 - \left(5 + (10-5) \times \left(\frac{f_{type} - LSL_i}{T_i - LSL_i}\right)^s\right)\right) \times \frac{f_{bonus}}{10} \times \alpha & \\ 5 & \text{if } f_{type} < LSL_i \end{cases}$$

Desirability function terms

As stated in Chapter 4, for this thesis, a priori preference elicitation is most suitable. A priori elicitation was chosen due to the recommendatory nature of the output of the MCO model: a priori elicitation of preferences would mean that the suggestions are more likely to be accurate from the beginning.

A priori preference elicitation was done via an interview and a questionnaire: the interview provided preference information for the desirability function terms (e.g. T, LSL, USL, and s for convexity) whereas the questionnaire featured questions pertaining the relative importance of the objective, used in *Step 6*. The resulting values, used in the visualisations of the following section, were calculated using a simple weighted-average of the number of responses.

Resulting desirability functions

The function terms for d_{time} are found in the legend of Figure 5.2, after transforming f_{time} from seconds to hours. Continuing, the function form and values for $d_{general}$ are thuswise seen in Figure 5.3. Lastly, the linear representation of f_{bonus}, using $\beta = 5$ is plotted and visualised in Figure 5.4.

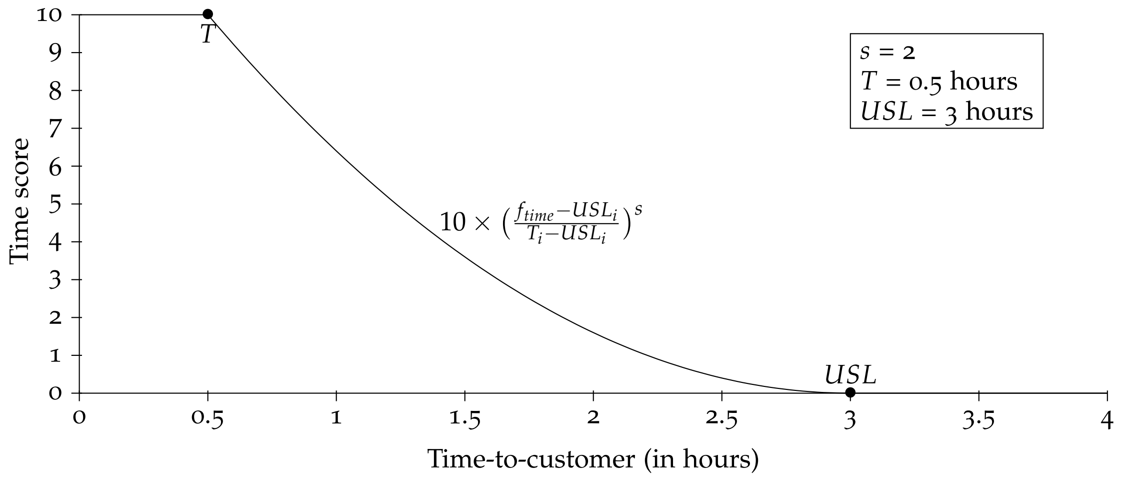


Figure 5.2: d_{time} visualised

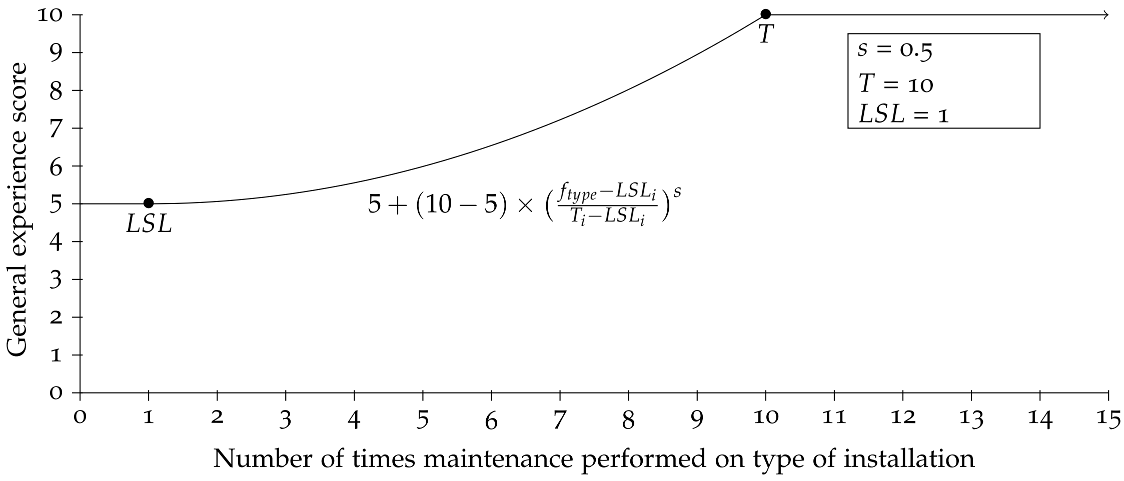


Figure 5.3: $d_{general}$ visualised

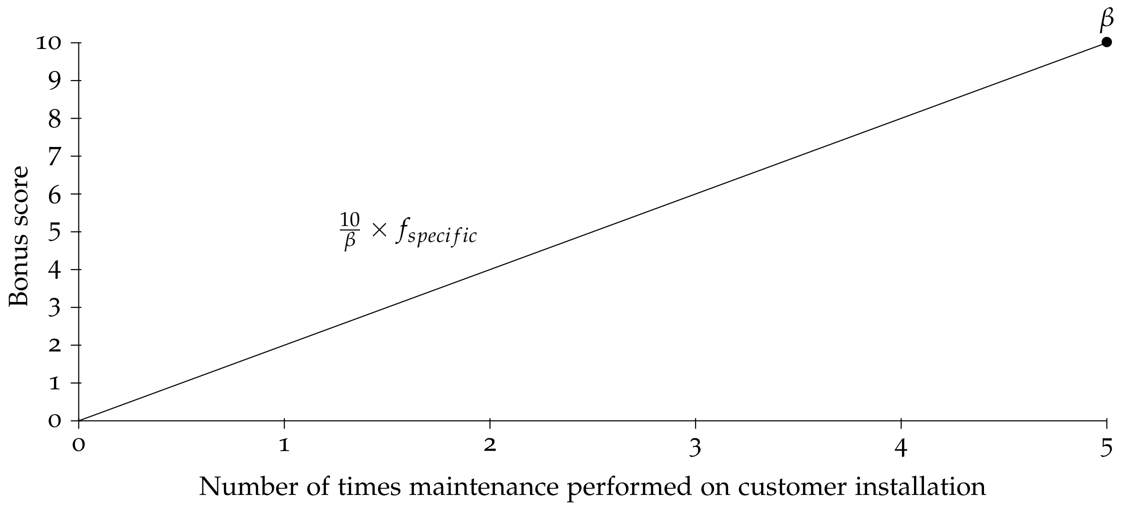


Figure 5.4: f_{bonus} visualised

Step 4: Normalise to grading scale

By design, desirability functions need no transformation and are already commensurable: standard desirability functions map to $[0, 1]$ and the modified desirability functions d_{time} and $d_{experience}$ map to $[0, 10]$, see *Step 3*.

Step 5: Determine importance of each objective

Following the interviews a short three-part questionnaire was used to determine the relative importance of both objectives. The questionnaire consisted of three questions, regarding service engineer assignment by dispatchers:

Question	Strongly disagree	Disagree	Somewhat agree	Agree	Strongly agree
Travel time to customer is important.	○	○	○	○	○
The amount of experience a service engineer has with a particular installation is important.	○	○	○	○	○
How often a service engineer has performed a specific task is important.	○	○	○	○	○

Normally the weights computed are of the form $w = [1, \infty]^m$. To compare two weights with numeric values of that form, relative importance is not explicit and hard to interpret. Preference elicitation and communicating back to the DM was deemed to be easier when the weights explicitly represent their relative importance by fractions of their sum. This removes ambiguity by expressing preferences in the same way. In addition, the weights are decidedly $w_i \in [0, 1]$ and $\sum_{i=1}^{m} w_i = 1$ to ensure the desirability index remains within the stated bounds, which the DM is familiar with. Extracting responses from the questionnaire, using

$$\frac{\overline{response_i}}{\sum_{i=1}^{m} \overline{response_i}}$$

yielded $w_{experience} = 0.47$ and $w_{time} = 0.53$.

Step 6: Scalarisation

Generally, scalarisation is achieved by computing the DI, desirability function approaches, after evaluating the corresponding desirability functions:

$$DI = \sqrt[m]{\prod_{i=1}^{m} d_i(f_i(x_1, x_2, \ldots, x_m)))}$$

In this context, because of the design choices regarding the grading scale and weights, ensuring that the DI stays within $[0, 10]$, a modified surprisingly simplified formula was devised:

$$Desirability\ index(DI) = d_{time}^{w_{time}} \times d_{eperience}^{w_{experience}}$$

Step 7: Evaluation

Computing the overall desirability function for every available service engineer, regarding an unassigned notification, allows for ranking based on the desirability index numeric. Recommended assignment is done solely by sorting on the highest values and listing the information of the corresponding service engineers, see example 3, Chapter 2.

In a practical sense a design choice was made to differentiate between generally sufficiently suitable and insufficiently suitable: sufficient marks for both the desirability functions and desirability index are all greater than or equal to five. Recommendations will only feature insufficient assignments if there are no sufficient assignments (DI < 5). Additionally, infeasible assignments (DI = 0) will only be shown if there are no feasible assignments. Dispatchers will have to review the infeasible assignments and pick the best.

5.2 Formulae summary

To reiterate, the following equations are used in the implementation. Note: T is the target value, LSL lower specified limit, USL upper specified limit, f the objective function values., and α the maximum specified percentage of the remainder to be alloted as bonus.

Time-to-customer is given by:

$$Time\text{-}to\text{-}customer\ (d_{time}) = \begin{cases} 10 & \text{if } f_{time} < T_i \\ 10 \times \left(\frac{f_{time} - USL_i}{T_i - USL_i}\right) & \text{if } T_i \le f_{time} \le USL_i \\ 0 & \text{if } f_{time} > USL_i \end{cases}$$

Experience desirability function $d_{experience}$ is given by:

$$Experience(d_{experience}) = \begin{cases} 10 & \text{if } f_{type} \ge T_i \\ \text{General score: } 5 + (10-5) \times \left(\frac{f_{type} - LSL_i}{T_i - LSL_i}\right)^s & \text{if } LSL_i \le f_{type} \le T_i \\ \text{Bonus: } \left(10 - \left(5 + (10-5) \times \left(\frac{f_{type} - LSL_i}{T_i - LSL_i}\right)^s\right)\right) \times \frac{f_{bonus}}{10} \times \alpha & \\ 5 & \text{if } f_{type} < LSL_i \end{cases}$$

Lastly, scalarisation with modified weights,while adhering to the Dutch grading scale, was done using:

$$Desirability\ index(DI) = d_{time}^{w_{time}} \times d_{eperience}^{w_{experience}}$$

Chapter 6

Implementation

As stated in Section 4.2, the model is implemented via a KNIME workflow. In addition, a dashboard similar to the existing dashboard for preventive maintenance, is implemented using Tableau.

6.1 Abstract workflow

Designing a system based on the structured approach of Section Chapter 5, started by drawing the perceived data flow, for the KNIME workflow, in business process model notation (BPMN). See Appendix B.

6.2 KNIME workflow

Above abstract workflow was implemented in five steps. See Appendix C for bigger figures.

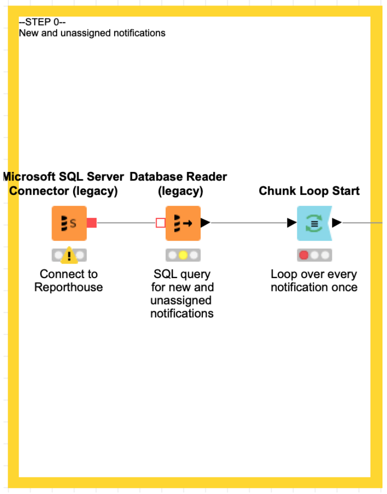

--STEP 0--
New and unassigned notifications
Microsoft SQL Server Connector (legacy)
Database Reader (legacy)
Chunk Loop Start
Connect to Reporthouse
SQL query for new and unassigned notifications
Loop over every notification once

Figure 6.1: Step 0 in KNIME

Step 0: Get all unassigned notifications

Dubbed step zero because it is preprocessing of the model, not necessarily part of the model: the input data, all unassigned notification, is loaded and the loop (i.e. the model) over these notifications started.

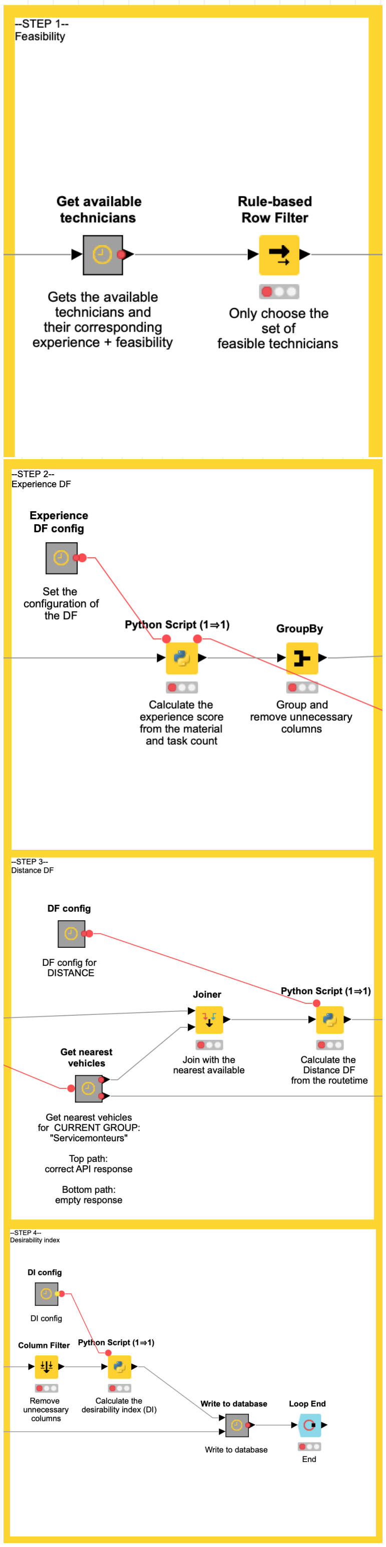

--STEP 1--
Feasibility
Get available technicians
Rule-based Row Filter
Gets the available technicians and their corresponding experience + feasibility
Only choose the set of feasible technicians
--STEP 2--
Experience DF
Experience DF config
Set the configuration of the DF
Python Script (1⇒1)
GroupBy
Calculate the experience score from the material and task count
Group and remove unnecessary columns
--STEP 3--
Distance DF
DF config
DF config for DISTANCE
Joiner
Python Script (1⇒1)
Join with the nearest available
Calculate the Distance DF from the routetime
Get nearest vehicles
Get nearest vehicles for CURRENT GROUP: "Servicemonteurs"
Top path: correct API response
Bottom path: empty response
--STEP 4--
Desirability index
DI config
DI config
Column Filter
Python Script (1⇒1)
Remove unnecessary columns
Calculate the desirability index (DI)
Write to database
Loop End
Write to database
End

Figure 6.2: Step 1-4 in KNIME

Step 1: Feasibility check

In this step in the first node, currently available service engineers data rows are joined with their experience matrix. Secondly, their corresponding feasibility, for the installation type of the current task in the loop, is checked. Only feasible service engineers go through to the next step.

Step 2: Experience calculation

Using the function term values, calculated in *Step* 3 of Section 5.1, as the desirability function configuration, in the middle node, using Python 3, the experience score $d_{experience}$ is calculated for each service engineer.

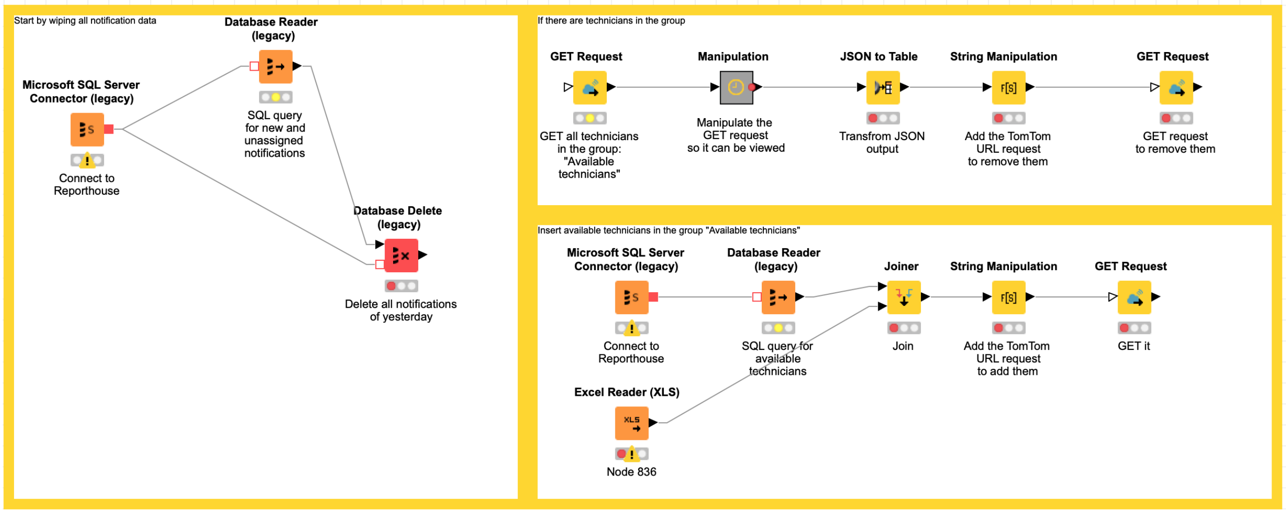
Step 3: Time-to-customer calculation

This step features the second desirability function configuration node. Below that, within a metanode, the distance from each available technician to a customer location is calculated using the WEBFLEET.connect API. These distances are then joined with the existing data rows and used in the calculation of d_{time}. WEBFLEET.connect sometimes returns error codes and incorrect responses, in these cases the next step is skipped.

Step 4: Desirability index

As for the last step, the desirability index DI is calculated and written to a database, where the loop ends and the next notification is run through the loop.

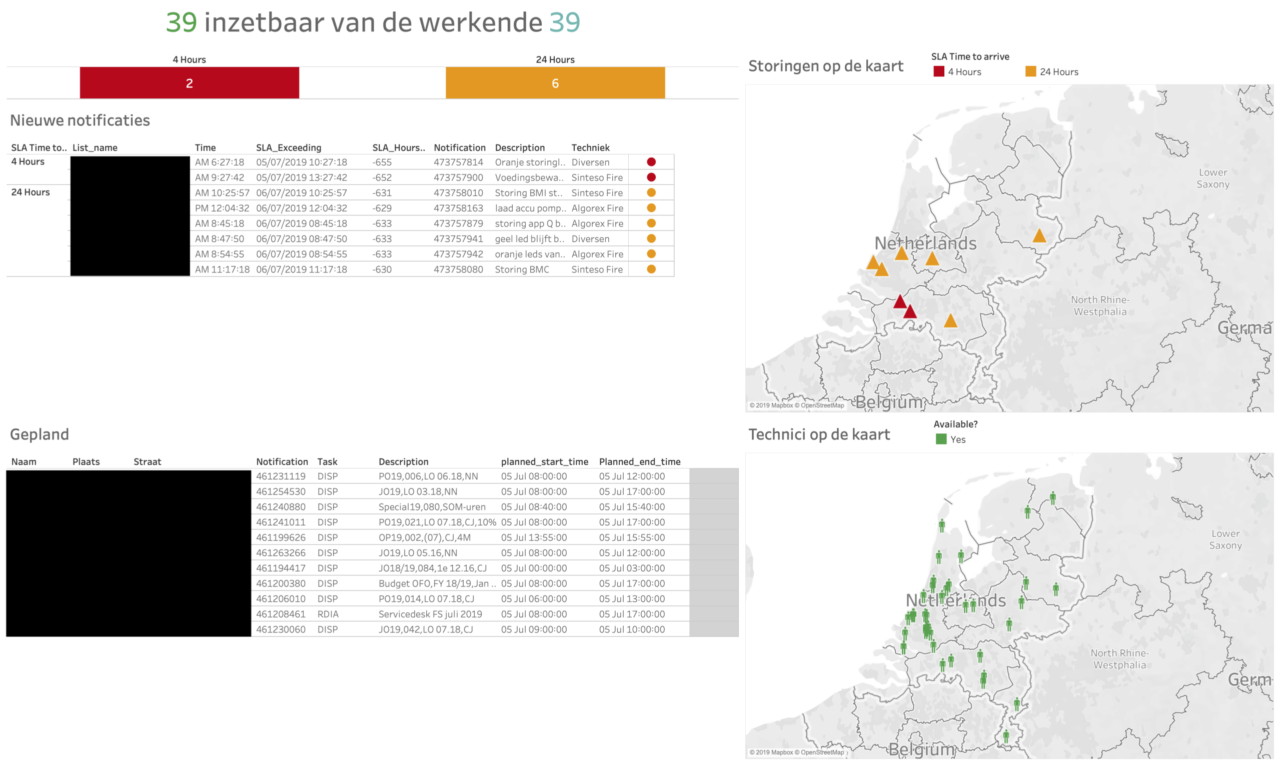
Lastly, to ensure that everything is set up properly for the main workflow, a secondary workflow was designed: this workflow is run daily in the morning for removal of old recommendations in the database and loads that day's available service engineers in the virtual group via the WEBFLEET.connect API.



Start by wiping all notification data
Microsoft SQL Server Connector (legacy)
Connect to Reporthouse
Database Reader (legacy)
SQL query for new and unassigned notifications
Database Delete (legacy)
Delete all notifications of yesterday
If there are technicians in the group
GET Request
GET all technicians in the group: "Available technicians"
Manipulation
Manipulate the GET request so it can be viewed
JSON to Table
Transfrom JSON output
String Manipulation
Add the TomTom URL request to remove them
GET Request
GET request to remove them
Insert available technicians in the group "Available technicians"
Microsoft SQL Server Connector (legacy)
Connect to Reporthouse
Database Reader (legacy)
SQL query for available technicians
Excel Reader (XLS)
Node 836
Joiner
Join
String Manipulation
Add the TomTom URL request to add them
GET Request
GET it

Figure 6.3: Left: removal of old recommendations. Right top/bottom: removal of old group and loading of that day's available service engineers

6.3 Tableau dashboard



Nieuwe notificaties

SLA Time to..	List_name	Time	SLA_Exceeding	SLA_Hours..	Notification	Description	Techniek	
4 Hours	[redacted]	AM 6:27:18	05/07/2019 10:27:18	-655	473757814	Oranje storingl..	Diversen	●
		AM 9:27:42	05/07/2019 13:27:42	-652	473757900	Voedingsbewa..	Sinteso Fire	●
24 Hours		AM 10:25:57	06/07/2019 10:25:57	-631	473758010	Storing BMI st..	Sinteso Fire	●
		PM 12:04:32	06/07/2019 12:04:32	-629	473758163	laad accu pomp..	Algorex Fire	●
		AM 8:45:18	06/07/2019 08:45:18	-633	473757879	storing app Q b..	Algorex Fire	●
		AM 8:47:50	06/07/2019 08:47:50	-633	473757941	geel led blijft b..	Diversen	●
		AM 8:54:55	06/07/2019 08:54:55	-633	473757942	oranje leds van..	Algorex Fire	●
		AM 11:17:18	06/07/2019 11:17:18	-630	473758080	Storing BMC	Sinteso Fire	●

Gepland

Naam	Plaats	Straat	Notification	Task	Description	planned_start_time	Planned_end_time
[redacted]			461231119	DISP	PO19,006,LO 06.18,NN	05 Jul 08:00:00	05 Jul 12:00:00
			461254530	DISP	JO19,LO 03.18,NN	05 Jul 08:00:00	05 Jul 17:00:00
			461240880	DISP	Special19,080,SOM-uren	05 Jul 08:40:00	05 Jul 15:40:00
			461241011	DISP	PO19,021,LO 07.18,CJ,10%	05 Jul 08:00:00	05 Jul 17:00:00
			461199626	DISP	OP19,002,(07),CJ,4M	05 Jul 13:55:00	05 Jul 15:55:00
			461263266	DISP	JO19,LO 05.16,NN	05 Jul 08:00:00	05 Jul 12:00:00
			461194417	DISP	JO18/19,084,1e 12.16,CJ	05 Jul 00:00:00	05 Jul 03:00:00
			461200380	DISP	Budget OFO,FY 18/19,Jan ..	05 Jul 08:00:00	05 Jul 17:00:00
			461206010	DISP	PO19,014,LO 07.18,CJ	05 Jul 06:00:00	05 Jul 13:00:00
			461208461	RDIA	Servicedesk FS juli 2019	05 Jul 08:00:00	05 Jul 17:00:00
			461230060	DISP	JO19,042,LO 07.18,CJ	05 Jul 09:00:00	05 Jul 10:00:00

Figure 6.4: Obfuscated view of the dashboard (See Appendix D for a bigger view)

Chapter 7

Results and conclusions

Concluding on the success of answering the research question

> *How to best implement and design a model for real-time assignment of service engineers based on multiple criteria optimisation?*

firstly, to summarise, the main results:

— An experience matrix was constructed

— Modified scoring functions were designed for the objectives *Time-to-customer* and *Experience*

— A KNIME workflow was developed, implementing the scoring functions. Scoring is done on real-time data and the recommendatory scores, as output, written to a database, to be shown on a dashboard for dispatchers

— Using Tableau, a dashboard was constructed visualising real-time data and the recommendations

Postive results were: model performance is promising for further development: during testing, recommendations, for a single notification, were computed in, on average, 35 seconds. In the future on a server, computation will be significantly faster and continuous up-to-date recommendations can be served on the dashboard, because of the low number of open notifications. Secondly, the model has a solid theoretical foundation for incremental development; adding objectives and drawing new desirability functions is easy because of the high modularity.

Results to improve: During an interview with a dispatcher it became apparent that many exceptions are not incorporated in the model and in turn the recommended assignments. Ergo, the current solution is to approach each notification in a too generalised fashion. However, adding objectives is possible and could remedy this.

Currently, full bonus to $d_{experience}$ is awarded after five service calls on a particular customer installation. A full bonus theoretically means a perfect assignment with regards to the experience objective. Conclusion is that at the moment, bonuses are awarded too freely and frequently. Additionaly, for general experience, $d_{general}$, target value T is set at ten, which also frequently equates to a perfect assignment among all service engineers.

Lastly, for further development or trying a similar project, it is highly recommended to heed next Chapter's notions and conclusions above!

Chapter 8

Further research & recommendations

As concluded, the model is too general, further development is recommended to account for exceptions and irregularities which would result in a model, better fitting reality. It is supposed that computing a complexity measure for each task or per notification, in combination with context variables such as the number of available and feasible service engineers, will improve the model accuracy.

Scoring on experience can cause dispatchers to (unconciously) discriminate more on experience than necessary. For more complex tasks it would be very tempting to repeatedly send the most experienced service engineer. This would however pose a serious business risk: other service engineers would miss out on valuable experience with a complex system which could prove troublesome in the future and most importantly, it could create single points of knowledge (SPOK). If such service engineers were to leave it would lead to a massive knowledge/experience loss for the whole division. Moreover, it is better to 'spread' experience and know-how between all service engineers not only for SPOK elimination, but BT division benefits from a more equally experienced pool of service engineers. Lastly, always sending a particular service engineer familiar with a particular customer installation could overlook irregularities because of his familiarity. It could prove to be more costly in the long-run, as the service engineer would be quick to fix the defect but glance over peculiarities that an unfamiliar service engineer would question [Sie19b] . Therefore, load balancing, in addition possibly SPOK elimination, is beneficial to further research and ultimately incorporate.

Since every notification is treated in an equal and generalised way, the system foregoes on one particular dimension of the nature of notifications for corrective maintenance: urgency. Further research is necessary to incorporate a measure for urgency into the model; modelling urgency could be done in an interactive way by allowing dispatcher input or via an extended version of perhaps the complexity feature proposed above. This allows for extensive further research.

Full bonus marks are awarded to $d_{experience}$ after five service calls on a particular customer installation, without regarding what kind of maintenance was performed on the installation; there are a plethora of distinct tasks and different defects possible. Current configuration of experience scoring can be extended by adding a measure of specific task experience; this would effectively add another layer in the hierarchic objective.

Additionally, experience is now calculated in the experience as an absolute value per individual. It could prove better for dispatchers to distinguish on experience if the score is calculated taking only into account the experience of the feasible set. Relative scoring could be implemented.

Currently all assignments are calculated using a greedy approach by computing the best assignment for every notification individually. This is possible in this context with few notifications concurrently, however, the approach yields lower overall accuracy in situations with large amounts of notifications to be assigned at high velocities. It could prove beneficial to add simultaneous assignment of all unassigned notifications; the capability to assign using batches will optimise not individual assignments but the overall assignments: combinatorial optimisation could for example save fuel, load balancing could spread service calls over more service engineers, ensuring that the most experienced service engineer is not always overly busy. The current model already supports greedy simultaneous assignment, but not other forms and optimisations like simulated annealing [VLA87]. Ergo, there is merit in further research into simultaneous assignment and potential benefits using more complex optimisation approaches.

As the model is further developed and confidence in the assignments grows, autonomous scheduling could be implemented. As the scheduler for preventive maintenance has this functionality, both systems could be combined, its functionality could be incorporated, or the model could delegate by sending scheduling orders to the automated scheduler. It should be noted that experimental results do not always agree with practical situations because of external unmodelled influences. Thus, for the foreseeable future, it is recommended that dispatchers regularly inspect each recommended assignment and treat it as such, a recommendation. Inspections will become even more important if or when the system ever becomes autonomous. Developing tests, sample checks, and or inspections are viable subjects for research.

Bibliography

[AG19] KNIME AG. About knime - knime, https://www.knime.com/about, referenced: 2019-06-28, 2019.

[CLP11] Nuno R Costa, João Lourenço, and Zulema L Pereira. Desirability function approach: a review and performance evaluation in adverse conditions. *Chemometrics and Intelligent Laboratory Systems*, 107(2):234–244, 2011.

[DS80] George Derringer and Ronald Suich. Simultaneous optimization of several response variables. *Journal of quality technology*, 12(4):214–219, 1980.

[GEF10] Salvatore Greco, Matthias Ehrgott, and José Rui Figueira. *Trends in multiple criteria decision analysis*, volume 142. Springer Science & Business Media, 2010.

[Har65] Edwin C Harrington. The desirability function. *Industrial quality control*, 21(10):494–498, 1965.

[HG12] Raphael T Haftka and Zafer Gürdal. *Elements of structural optimization*, volume 11. Springer Science & Business Media, 2012.

[JRFE05] S. Greco J. R. Figueira and M. Ehrgott. *Multiple criteria decision analysis*. New York: Springer, 2005.

[Kee74] Ralph L. Keeney. Multiplicative utility functions. *Operations Research*, 22(1):22–34, 1974.

[KGC13] Prasad Karande, Susanta Kumar Gauri, and Shankar Chakraborty. Applications of utility concept and desirability function for materials selection. *Materials & Design*, 45:349–358, 2013.

[KJ19] Max Kuhn and Kjell Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. CRC Press, 2019.

[KM04] John F Kros and Christina M Mastrangelo. Comparing multi-response design methods with mixed responses. *Quality and Reliability Engineering International*, 20(5):527–539, 2004.

[KR93] Ralph L Keeney and Howard Raiffa. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press, 1993.

[KRWR79] Ralph Keeney, Howard Raiffa, and David W. Rajala. Decisions with multiple objectives: Preferences and value trade-offs. *Systems, Man and Cybernetics, IEEE Transactions on*, 9:403 – 403, 08 1979.

[MA04] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[Mie98] K. Miettinen. *Nonlinear Multiobjective Optimization*. Boston, MA: Springer US, 1998.

[MT06] J Mehnen and H Trautmann. Integration of experts preferences in pareto optimization by desirability function techniques. *parameters*, 500:1, 2006.

[Nuf18] Nuffic. Grading systems in the netherlands, the united states and the united kingdom, https://www.studyinholland.nl/documentation/grading-systems-in-the-netherlands-the-united-states-and-the-united-kingdom.pdf, referenced: 2019-07-27, 2018.

[Sie19a] Siemens. https://new.siemens.com/global/en/company/sustainability.html, referenced: 2019-06-24, 2019.

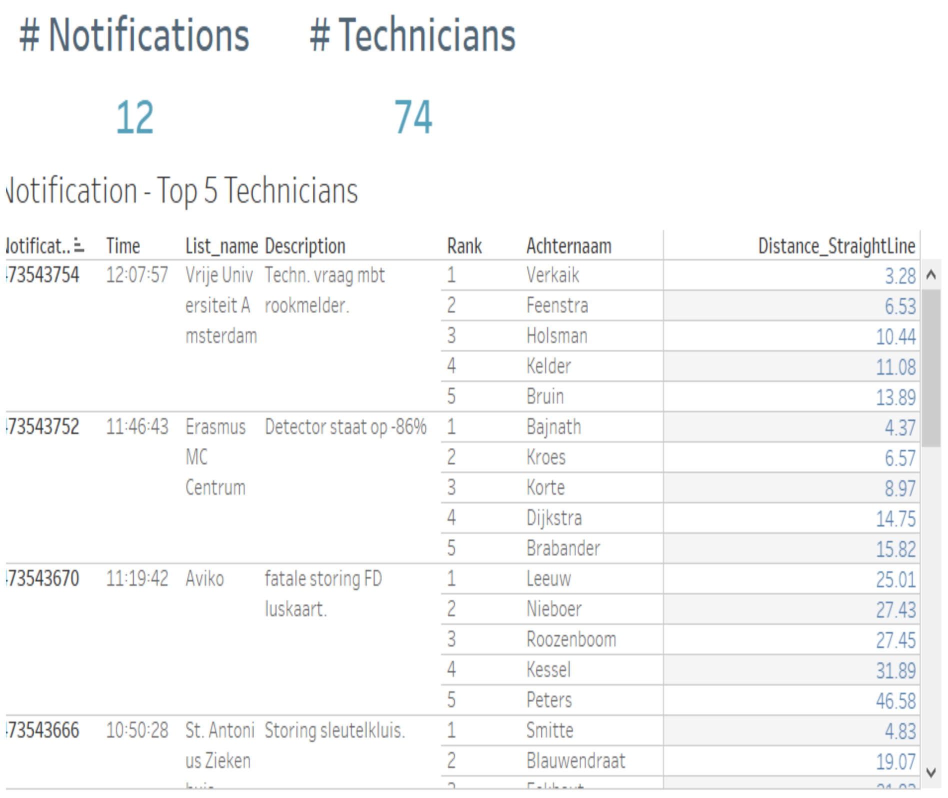
[Sie19b] Siemens. Said during an interview, 2019.

[Ste99] Detlef Steuer. Multi-criteria-optimisation and desirability indices. *Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten* , 1999.

[VLA87] Peter JM Van Laarhoven and Emile HL Aarts. *Simulated annealing*. Springer, 1987.

Appendix A

MVP



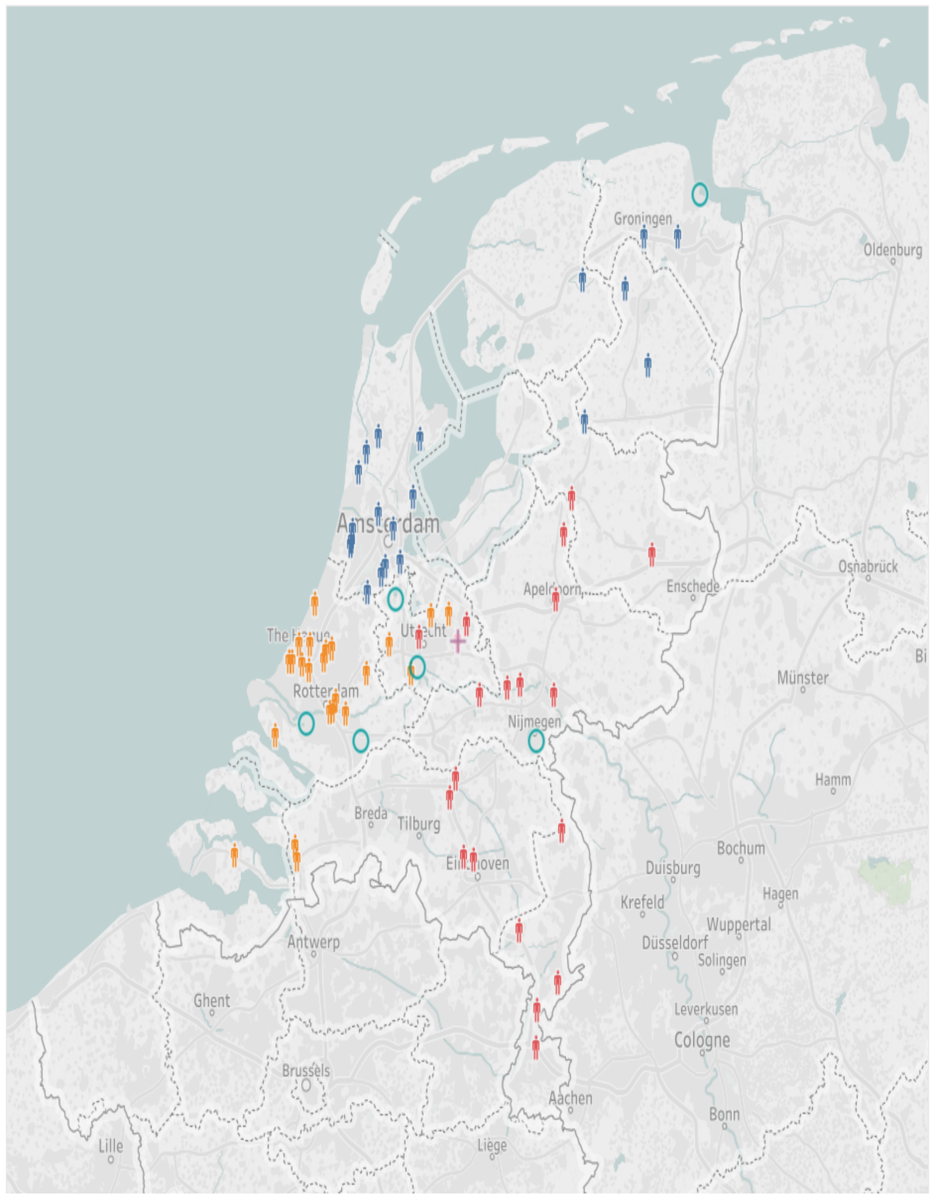
Notifications # Technicians

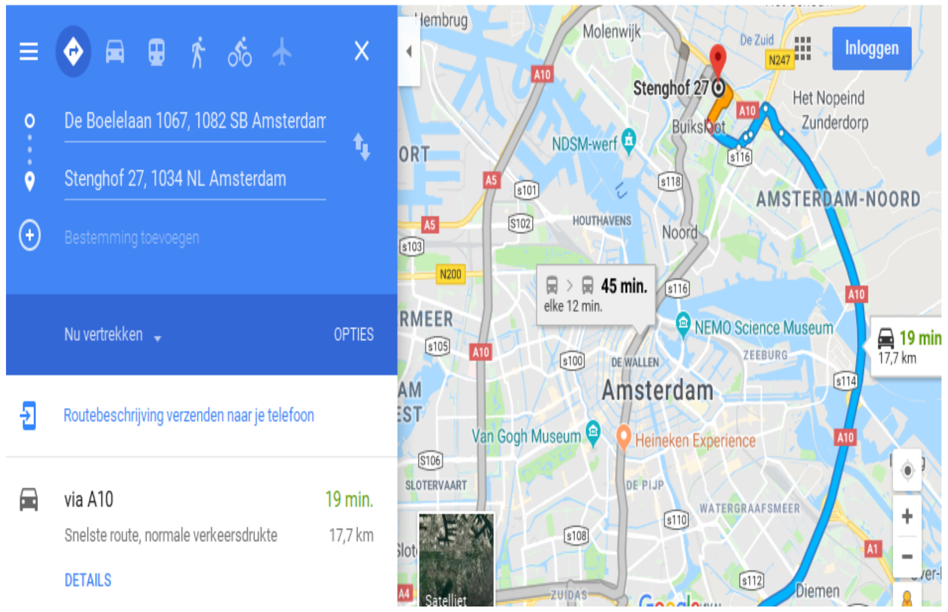
12 74

Notification - Top 5 Technicians

Notificat..	Time	List_name	Description	Rank	Achternaam	Distance_StraightLine
73543754	12:07:57	Vrije Universiteit Amsterdam	Techn. vraag mbt rookmelder.	1	Verkaik	3.28
				2	Feenstra	6.53
				3	Holsman	10.44
				4	Kelder	11.08
				5	Bruin	13.89
73543752	11:46:43	Erasmus MC Centrum	Detector staat op -86%	1	Bajnath	4.37
				2	Kroes	6.57
				3	Korte	8.97
				4	Dijkstra	14.75
				5	Brabander	15.82
73543670	11:19:42	Aviko	fatale storing FD luskaart.	1	Leeuw	25.01
				2	Nieboer	27.43
				3	Roozenboom	27.45
				4	Kessel	31.89
				5	Peters	46.58
73543666	10:50:28	St. Antonius Zieken	Storing sleutelkluis.	1	Smitte	4.83
				2	Blauwendraat	19.07

Notifications vs Technicians




De Boelelaan 1067, 1082 SB Amsterdam

Stenghof 27, 1034 NL Amsterdam

Bestemming toevoegen

Nu vertrekken OPTIES

Routebeschrijving verzenden naar je telefoon

via A10 19 min.

Snelste route, normale verkeersdrukte 17,7 km

DETAILS

Figure A.1: Minimal viable product mock-up

Appendix B

Abstract workflow

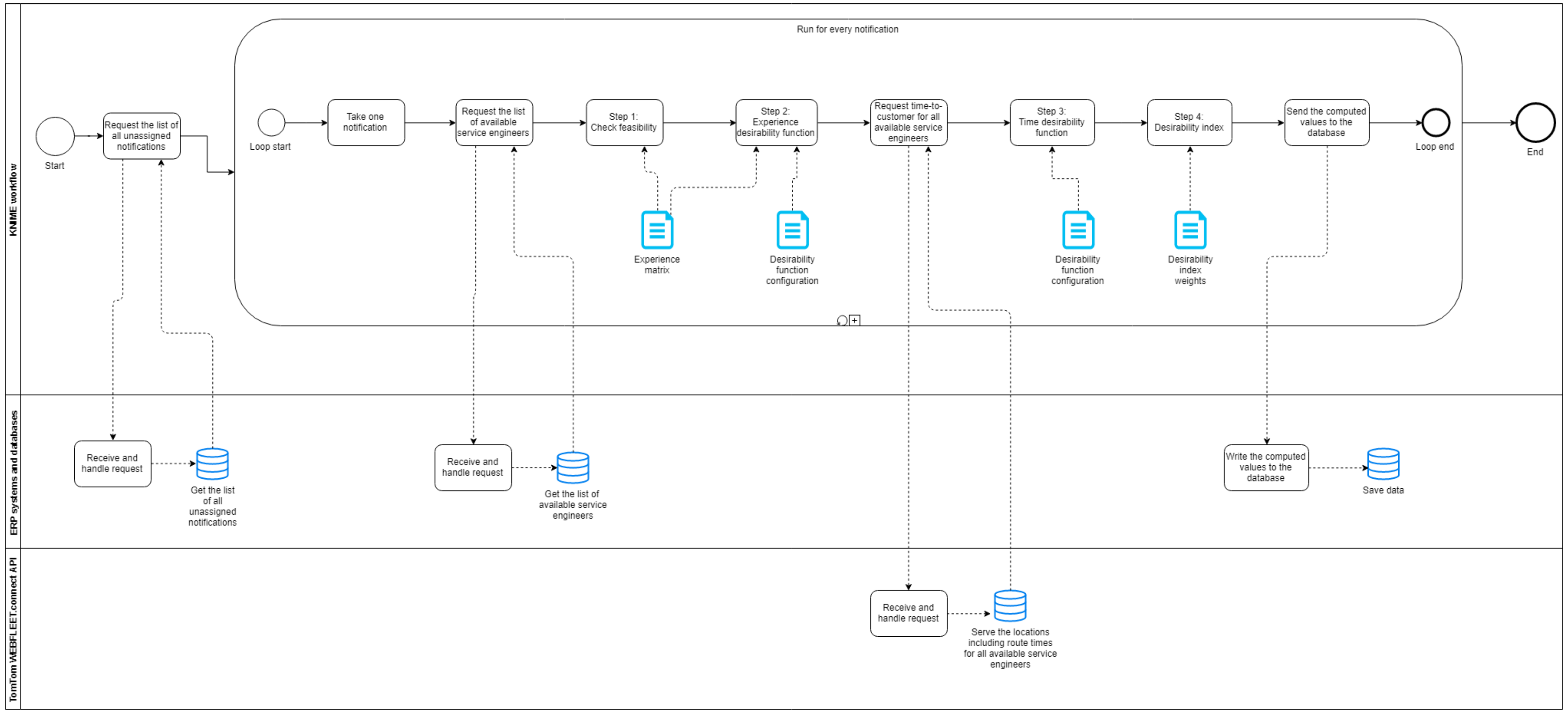
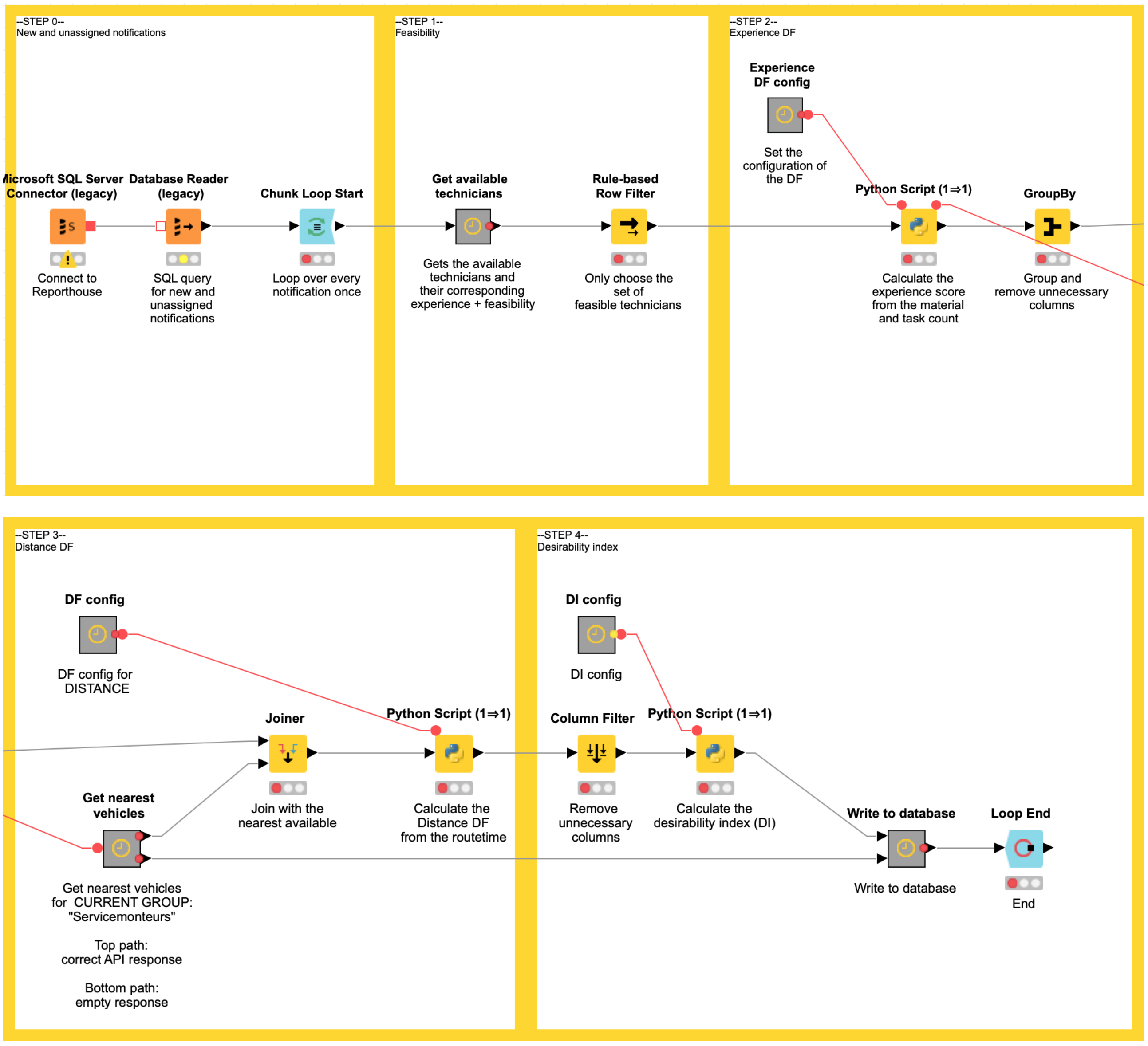

KNIME workflow
ERP systems and databases
TomTom WEBFLEET.connect API
Start
Request the list of all unassigned notifications
Run for every notification
Loop start
Take one notification
Request the list of available service engineers
Step 1: Check feasibility
Step 2: Experience desirability function
Request time-to-customer for all available service engineers
Step 3: Time desirability function
Step 4: Desirability index
Send the computed values to the database
Loop end
End
Experience matrix
Desirability function configuration
Desirability function configuration
Desirability index weights
Receive and handle request
Get the list of all unassigned notifications
Receive and handle request
Get the list of available service engineers
Write the computed values to the database
Save data
Receive and handle request
Serve the locations including route times for all available service engineers

Figure B.1: Abstract workflow

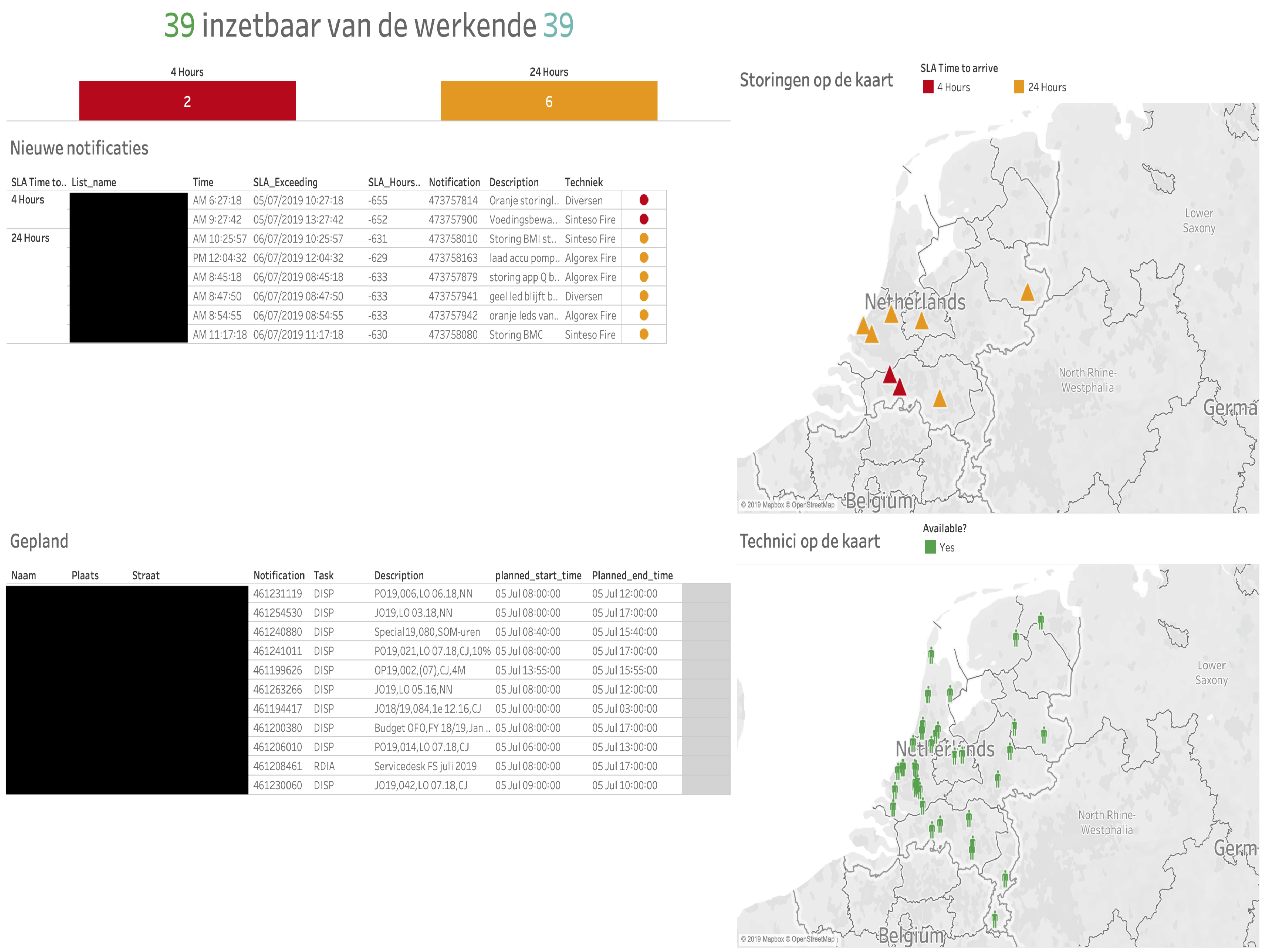
Appendix C

KNIME workflow



--STEP 0--
New and unassigned notifications
Microsoft SQL Server Connector (legacy)
Connect to Reporthouse
Database Reader (legacy)
SQL query for new and unassigned notifications
Chunk Loop Start
Loop over every notification once
--STEP 1--
Feasibility
Get available technicians
Gets the available technicians and their corresponding experience + feasibility
Rule-based Row Filter
Only choose the set of feasible technicians
--STEP 2--
Experience DF
Experience DF config
Set the configuration of the DF
Python Script (1⇒1)
Calculate the experience score from the material and task count
GroupBy
Group and remove unnecessary columns
--STEP 3--
Distance DF
DF config
DF config for DISTANCE
Joiner
Join with the nearest available
Python Script (1⇒1)
Calculate the Distance DF from the routetime
Get nearest vehicles
Get nearest vehicles for CURRENT GROUP: "Servicemonteurs"
Top path: correct API response
Bottom path: empty response
--STEP 4--
Desirability index
DI config
DI config
Column Filter
Remove unnecessary columns
Python Script (1⇒1)
Calculate the desirability index (DI)
Write to database
Write to database
Loop End
End

Figure C.1: The implemented KNIME workflow



39 inzetbaar van de werkende 39

4 Hours	24 Hours
2	6

Nieuwe notificaties

SLA Time to..	List_name	Time	SLA_Exceeding	SLA_Hours..	Notification	Description	Techniek	
4 Hours	[redacted]	AM 6:27:18	05/07/2019 10:27:18	-655	473757814	Oranje storingl..	Diversen	●
	[redacted]	AM 9:27:42	05/07/2019 13:27:42	-652	473757900	Voedingsbewa..	Sinteso Fire	●
24 Hours	[redacted]	AM 10:25:57	06/07/2019 10:25:57	-631	473758010	Storing BMI st..	Sinteso Fire	●
	[redacted]	PM 12:04:32	06/07/2019 12:04:32	-629	473758163	laad accu pomp..	Algorex Fire	●
	[redacted]	AM 8:45:18	06/07/2019 08:45:18	-633	473757879	storing app Q b..	Algorex Fire	●
	[redacted]	AM 8:47:50	06/07/2019 08:47:50	-633	473757941	geel led blijft b..	Diversen	●
	[redacted]	AM 8:54:55	06/07/2019 08:54:55	-633	473757942	oranje leds van..	Algorex Fire	●
	[redacted]	AM 11:17:18	06/07/2019 11:17:18	-630	473758080	Storing BMC	Sinteso Fire	●

Storingen op de kaart

SLA Time to arrive
🟥 4 Hours
🟧 24 Hours

Gepland

Naam	Plaats	Straat	Notification	Task	Description	planned_start_time	Planned_end_time
[redacted]	[redacted]	[redacted]	461231119	DISP	P019,006,LO 06.18,NN	05 Jul 08:00:00	05 Jul 12:00:00
[redacted]	[redacted]	[redacted]	461254530	DISP	J019,LO 03.18,NN	05 Jul 08:00:00	05 Jul 17:00:00
[redacted]	[redacted]	[redacted]	461240880	DISP	Special19,080,SOM-uren	05 Jul 08:40:00	05 Jul 15:40:00
[redacted]	[redacted]	[redacted]	461241011	DISP	P019,021,LO 07.18,CJ,10%	05 Jul 08:00:00	05 Jul 17:00:00
[redacted]	[redacted]	[redacted]	461199626	DISP	OP19,002,(07),CJ,4M	05 Jul 13:55:00	05 Jul 15:55:00
[redacted]	[redacted]	[redacted]	461263266	DISP	J019,LO 05.16,NN	05 Jul 08:00:00	05 Jul 12:00:00
[redacted]	[redacted]	[redacted]	461194417	DISP	J018/19,084,1e 12.16,CJ	05 Jul 00:00:00	05 Jul 03:00:00
[redacted]	[redacted]	[redacted]	461200380	DISP	Budget OFO,FY 18/19,Jan ..	05 Jul 08:00:00	05 Jul 17:00:00
[redacted]	[redacted]	[redacted]	461206010	DISP	P019,014,LO 07.18,CJ	05 Jul 06:00:00	05 Jul 13:00:00
[redacted]	[redacted]	[redacted]	461208461	RDIA	Servicedesk FS juli 2019	05 Jul 08:00:00	05 Jul 17:00:00
[redacted]	[redacted]	[redacted]	461230060	DISP	J019,042,LO 07.18,CJ	05 Jul 09:00:00	05 Jul 10:00:00

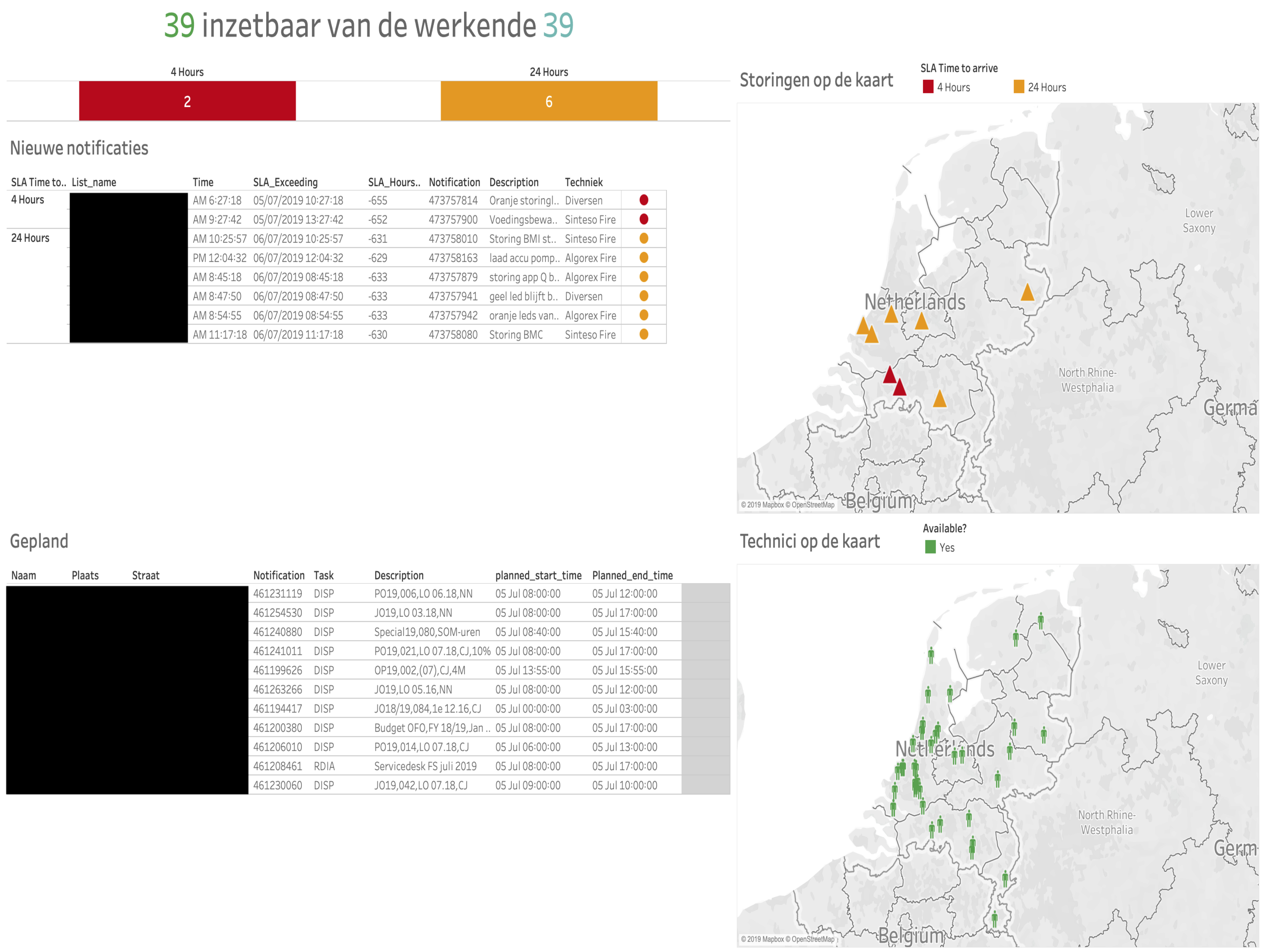
Technici op de kaart

Available?
🟩 Yes

Figure C.2: Obfuscated view of the dashboard

Appendix D

Dashboard



39 inzetbaar van de werkende 39

4 Hours	24 Hours
2	6

Nieuwe notificaties

SLA Time to..	List_name	Time	SLA_Exceeding	SLA_Hours..	Notification	Description	Techniek	
4 Hours	[redacted]	AM 6:27:18	05/07/2019 10:27:18	-655	473757814	Oranje storingl..	Diversen	●
	[redacted]	AM 9:27:42	05/07/2019 13:27:42	-652	473757900	Voedingsbewa..	Sinteso Fire	●
24 Hours	[redacted]	AM 10:25:57	06/07/2019 10:25:57	-631	473758010	Storing BMI st..	Sinteso Fire	●
	[redacted]	PM 12:04:32	06/07/2019 12:04:32	-629	473758163	laad accu pomp..	Algorex Fire	●
	[redacted]	AM 8:45:18	06/07/2019 08:45:18	-633	473757879	storing app Q b..	Algorex Fire	●
	[redacted]	AM 8:47:50	06/07/2019 08:47:50	-633	473757941	geel led blijft b..	Diversen	●
	[redacted]	AM 8:54:55	06/07/2019 08:54:55	-633	473757942	oranje leds van..	Algorex Fire	●
	[redacted]	AM 11:17:18	06/07/2019 11:17:18	-630	473758080	Storing BMC	Sinteso Fire	●

Storingen op de kaart

SLA Time to arrive
🟥 4 Hours 🟧 24 Hours

Gepland

Naam	Plaats	Straat	Notification	Task	Description	planned_start_time	Planned_end_time
[redacted]	[redacted]	[redacted]	461231119	DISP	P019,006,LO 06.18,NN	05 Jul 08:00:00	05 Jul 12:00:00
[redacted]	[redacted]	[redacted]	461254530	DISP	J019,LO 03.18,NN	05 Jul 08:00:00	05 Jul 17:00:00
[redacted]	[redacted]	[redacted]	461240880	DISP	Special19,080,SOM-uren	05 Jul 08:40:00	05 Jul 15:40:00
[redacted]	[redacted]	[redacted]	461241011	DISP	P019,021,LO 07.18,CJ,10%	05 Jul 08:00:00	05 Jul 17:00:00
[redacted]	[redacted]	[redacted]	461199626	DISP	OP19,002,(07),CJ,4M	05 Jul 13:55:00	05 Jul 15:55:00
[redacted]	[redacted]	[redacted]	461263266	DISP	J019,LO 05.16,NN	05 Jul 08:00:00	05 Jul 12:00:00
[redacted]	[redacted]	[redacted]	461194417	DISP	J018/19,084,1e 12.16,CJ	05 Jul 00:00:00	05 Jul 03:00:00
[redacted]	[redacted]	[redacted]	461200380	DISP	Budget OFO,FY 18/19,Jan ..	05 Jul 08:00:00	05 Jul 17:00:00
[redacted]	[redacted]	[redacted]	461206010	DISP	P019,014,LO 07.18,CJ	05 Jul 06:00:00	05 Jul 13:00:00
[redacted]	[redacted]	[redacted]	461208461	RDIA	Servicedesk FS juli 2019	05 Jul 08:00:00	05 Jul 17:00:00
[redacted]	[redacted]	[redacted]	461230060	DISP	J019,042,LO 07.18,CJ	05 Jul 09:00:00	05 Jul 10:00:00

Technici op de kaart

Available?
🟩 Yes

Figure D.1: Obfuscated view of the dashboard