

CRANFIELD UNIVERSITY

SCHOOL OF ENGINEERING

PhD THESIS

2009

D KLANN

The Role of Information Technology in the Airport Business:  
A Retail-Weighted Resource Management Approach  
for Capacity-Constrained Airports

Supervisor: Dr R. Pagliari

Academic Year 2003 to 2009

This thesis is submitted in partial fulfilment of the requirements  
for the degree of Doctor of Philosophy.

© Cranfield University, 2009. All rights reserved. No part of this publication may be reproduced without the written permission of the copyright holder.

Much research has been undertaken to gain insight into business alignment of IT. This alignment basically aims to improve a firm's performance by an improved harmonization of the business function and the IT function within a firm. The thesis discusses previous approaches and constructs an overall framework, which a potential approach needs to fit in.

Being in a highly regulated industry, for airports there is little space left to increase revenues. However, the retailing business has proven to be an area that may contribute towards higher income for airport operators. Consequently, airport management should focus on supporting this business segment. Nevertheless, it needs to be taken into account that smooth airport operations are a precondition for successful retailing business at an airport.

Applying the concept of information intensity, the processes of gate allocation and airport retailing have been determined to appraise the potential that may be realized upon (improved) synchronization of the two. It has been found that the lever is largest in the planning phase (i.e. prior to operations), and thus support by means of information technology (for information distribution and improved planning) may help to enable an improved overall retail performance.

In order to determine potential variables, which might influence the output, a process decomposition has been conducted along with the development of an appropriate information model.

The derived research model has been tested in different scenarios. For this purpose an adequate gate allocation algorithm has been developed and implemented in a purpose-written piece of software. To calibrate the model, actual data (several hundred thousand data items from Frankfurt Airport) from two flight plan seasons has been used.

Key findings: The results show that under the conditions described it seems feasible to increase retail sales in the magnitude of 9% to 21%. The most influential factors (besides the constraining rule set and a retail area's specific performance) proved to be a flight's minimum and maximum time at a gate as well as its buffer time at gate.

However, as some of the preconditions may not be accepted by airport management or national regulators, the results may be taken as an indication for cost incurred, in case the suggested approach is not considered.

The transferability to other airport business models and limitations of the research approach are discussed at the end along with suggestions for future areas of research.

**Keywords:**

gate assignment problem, retail sales, algorithm, combinatorial explosion, business process engineering, information intensity, data model, simulation framework.

|   |            |
|---|------------|
| ABSTRACT  | I          |
| LIST OF CONTENTS  | II         |
| LIST OF FIGURES   | IV         |
| LIST OF TABLES  | VI         |
| ACKNOWLEDGEMENTS  | VII        |
| ABBREVIATIONS   | VIII       |
| <br>  |            |
| <b>1 INTRODUCTION .....</b>   | <b>1</b>   |
| 1.1 General.....  | 2          |
| 1.2 Motivation.....   | 3          |
| 1.3 Thesis title.....   | 4          |
| 1.4 Aim and objective (demarcation).....  | 5          |
| 1.5 Potential contribution to knowledge .....                                   | 7          |
| 1.6 Organisation of this paper .....  | 8          |
| <b>2 SETTING THE SCENE .....</b>  | <b>9</b>   |
| 2.1 Success factors in the airport industry .....                               | 10         |
| 2.2 Alignment of business and information technology .....                      | 12         |
| 2.3 Gate allocation .....   | 24         |
| 2.3.1 Use of a resource management system (RMS).....                            | 28         |
| 2.4 Airport retailing .....   | 31         |
| 2.5 Integral view and research question .....                                   | 37         |
| <b>3 METHODOLOGY .....</b>  | <b>39</b>  |
| 3.1 Choice of methodology (Justification).....                                  | 40         |
| 3.2 Business context and scope.....   | 40         |
| 3.2.1 ‘Information Intensity’ model applied to airport business.....            | 42         |
| 3.2.2 Applicable airport environment.....                                       | 44         |
| 3.2.3 Business process decomposition for airport retailing .....                | 46         |
| 3.2.4 Business process decomposition for gate allocation.....                   | 49         |
| 3.2.5 Airport information model in context of business processes examined ..... | 51         |
| 3.3 Formulation of conceptual research model .....                              | 57         |
| 3.4 Identification of information / data needed.....                            | 60         |
| 3.4.1 Infrastructural (reference) data .....                                    | 61         |
| 3.4.2 Flight schedules .....  | 63         |
| 3.4.3 Retail sales data .....   | 64         |
| 3.4.4 Rule set.....   | 66         |
| 3.5 From conceptual research model towards a quantitative model .....           | 67         |
| 3.6 Approach to determine improved solutions .....                              | 70         |
| 3.6.1 Type of problem and mathematical considerations .....                     | 70         |
| 3.6.2 Heuristic approach versus deterministic approach .....                    | 75         |
| 3.6.3 Deterministic algorithm for retail-oriented gate allocation .....         | 77         |
| 3.7 Simulation environment.....   | 88         |
| 3.7.1 Components.....   | 89         |
| 3.7.2 Implementation.....   | 98         |
| <b>4 ANALYSIS .....</b>   | <b>102</b> |
| 4.1 Set of current flight data .....  | 103        |
| 4.1.1 Seasonality .....   | 103        |
| 4.1.2 Traffic distribution (passengers, flights) .....                          | 108        |
| 4.1.3 Retail area factors and sales figures (model calibration).....            | 110        |
| 4.1.4 Sales per passenger.....  | 117        |
| 4.1.5 Sales per flight.....   | 118        |
| 4.1.6 Comparison of retail sales (actual traffic vs. seasonal planning).....    | 119        |
| 4.1.7 Figures regarding flight operations .....                                 | 120        |

---

|          |  |            |
|----------|--|------------|
| 4.1.7.1  | Delay.....   | 121        |
| 4.1.7.2  | Turnaround times.....  | 122        |
| 4.1.7.3  | Gate changes versus changes in retail areas.....                               | 124        |
| 4.1.8    | Causality: Falsification using correlations.....                               | 125        |
| 4.1.9    | Summary: set of current flight data.....                                       | 126        |
| 4.2      | Scenario technique.....  | 127        |
| 4.2.1    | Elements describing a scenario.....  | 127        |
| 4.2.2    | Standard assumptions, rule set.....  | 128        |
| 4.2.3    | Scenarios.....   | 130        |
| 4.3      | Simulation runs.....   | 134        |
| 4.4      | Analysis of results.....   | 136        |
| 4.4.1    | Result of baseline scenario compared to actual result.....                     | 137        |
| 4.4.2    | Scenario results in order of retail sales result.....                          | 138        |
| <b>5</b> | <b>CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER RESEARCH.....</b>               | <b>146</b> |
| 5.1      | Recommended actions for airport managers.....                                  | 148        |
| 5.2      | The role of IT: Strategic implications for airport policy.....                 | 150        |
| 5.3      | Transferability.....   | 151        |
| 5.3.1    | Different players in same business context.....                                | 151        |
| 5.3.2    | Individual aspects to be considered.....                                       | 153        |
| 5.4      | Limitations of research undertaken.....  | 154        |
| 5.5      | Contributions.....   | 156        |
| 5.6      | Areas for future research.....   | 156        |
| <b>6</b> | <b>LIST OF REFERENCES.....</b>   | <b>159</b> |
| <b>7</b> | <b>BIBLIOGRAPHY.....</b>   | <b>166</b> |
| <b>8</b> | <b>APPENDIX A.....</b>   | <b>173</b> |
| 8.1      | Overview of functions and procedures.....                                      | 173        |
| 8.2      | Source code of software.....   | 174        |
| 8.2.1    | Main File.....   | 175        |
| 8.2.2    | Include File.....  | 278        |
| <b>9</b> | <b>APPENDIX B.....</b>   | <b>283</b> |
| 9.1      | Sample output of ABC data (raw).....   | 283        |
| 9.2      | Sample output of ABC data (formatted).....                                     | 284        |
| 9.3      | Sample output of simulation run (gate allocation, header).....                 | 285        |
| 9.4      | Sample output of simulation run (gate allocation, detail).....                 | 285        |
| 9.5      | Sample output of simulation run (flight schedule).....                         | 286        |
| 9.6      | Outline of report summary (sales, passengers per weekday and retail area)..... | 288        |
| 9.7      | Subset of full report (actual passenger distribution).....                     | 289        |
| 9.8      | Subset of full report (improved revenue distribution).....                     | 289        |
| 9.9      | Excerpt from summary report of simulation results.....                         | 290        |
| 9.10     | Simulation result: baseline scenario vs. actual traffic (passengers).....      | 291        |
| 9.11     | Simulation result: baseline scenario vs. actual traffic (sales).....           | 292        |

|  |     |
|--|-----|
| Figure 1: Factors of airport competitiveness. Source: Park (2003, p. 354). .....                             | 11  |
| Figure 2: Strategic alignment framework (for a global enterprise). Source: Peppard (1998, p. 10).....        | 12  |
| Figure 3: Strategic alignment according to Henderson and Venkatraman. [...].....                             | 13  |
| Figure 4: An IT investment framework. Adopted from Ross and Beath (2002). .....                              | 14  |
| Figure 5: Structure of balanced scorecard. Source: Kaplan and Norton (1992, p.72). .....                     | 15  |
| Figure 6: Impact of IT investments on productivity at industry level. Research model. [...]. .....           | 17  |
| Figure 7: Different architectural layers. (own illustration). .....  | 20  |
| Figure 8: The different IT architectural layers in context of their two dimensions. (own illustration). .... | 21  |
| Figure 9: Generic IT systems landscape at an airport according to an implementation scenario [...]. .....    | 22  |
| Figure 10: Research topic placed within body of knowledge (of IT alignment). .....                           | 23  |
| Figure 11: Model framework according to Hamzawi (1986, p. 193). .....  | 25  |
| Figure 12: Structured body of knowledge of the gate assignment problem (GAP). .....                          | 27  |
| Figure 13: Resource management system (effort distribution 1). .....   | 28  |
| Figure 14: Resource management system (effort distribution 2). .....   | 29  |
| Figure 15: Basic structure of Hildebrandt's (1988) 3-factor-model of store image [...]. .....                | 33  |
| Figure 16: Drivers to retail success - Control and impact. Source: Fraport AG. ....                          | 36  |
| Figure 17: Business context of research project. ....  | 41  |
| Figure 18: Value chain of an airport with focus on use of technology. ....                                   | 42  |
| Figure 19: Information intensity matrix, applied to airports. ....   | 43  |
| Figure 20: IDEF0 legend of an activity. ....   | 45  |
| Figure 21: Decomposition of retail process, top level. ....  | 46  |
| Figure 22: Decomposition of retail process, 2 <sup>nd</sup> level. ....                                      | 47  |
| Figure 23: Decomposition of gate allocation process, top level. ....   | 49  |
| Figure 24: Decomposition of gate allocation process, 2 <sup>nd</sup> level. ....                             | 50  |
| Figure 25: Resource Management Data Flow Chart (Source: Kelemen, 2005, p. 22). .....                         | 51  |
| Figure 26: Airport information model (outline) in context of research. ....                                  | 52  |
| Figure 27: Airport information model, domain: flight. ....   | 53  |
| Figure 28: Airport information model, domain: airport infrastructure. ....                                   | 54  |
| Figure 29: Airport information model, domain: flight event. ....   | 55  |
| Figure 30: Conceptual research model with underlying statement (H.1). ....                                   | 57  |
| Figure 31: Context of conceptual research model. ....  | 58  |
| Figure 32: Basic terminal layout, Frankfurt Airport. ....  | 62  |
| Figure 33: Timeline of flight plan seasons. ....   | 63  |
| Figure 34: Card-deck example transferred to gate allocation context (1). .....                               | 72  |
| Figure 35: Card-deck example transferred to gate allocation context (2). .....                               | 73  |
| Figure 36: Flowchart of a basic genetic algorithm. ....  | 75  |
| Figure 37: Proposed solution (framed branches) within GAP classification. ....                               | 77  |
| Figure 38: Algorithm for gate allocation, top level. ....  | 81  |
| Figure 39: Algorithm to compose an allocation (recursive search of solution space). ....                     | 86  |
| Figure 40: Basic architecture of simulation workbench. ....  | 89  |
| Figure 41: Data categories in simulation software, implemented as flat files. ....                           | 90  |
| Figure 42: Function tree of simulation software, top level. ....   | 91  |
| Figure 43: Functions for data cleansing. ....  | 92  |
| Figure 44: Functions to manage reference data. ....  | 94  |
| Figure 45: Functions to produce basic statistics (descriptive data). ....                                    | 95  |
| Figure 46: The heat map function. ....   | 95  |
| Figure 47: Screenshot of animated categorized data. ....   | 96  |
| Figure 48: Geographical representation of categorized data. ....   | 97  |
| Figure 49: Example of source code displayed within integrated development environment. ....                  | 99  |
| Figure 50: Sample display output during a simulation run. ....   | 100 |
| Figure 51: Formal seasonality in the research context. ....  | 104 |
| Figure 52: Seasonality within actual data (summer and winter). .....   | 105 |
| Figure 53: Seasonality within actual data (weeks in July). .....   | 106 |
| Figure 54: Seasonality within actual data (daily traffic waves). .....                                       | 107 |
| Figure 55: Distribution of passengers (sums) per retail area. ....   | 108 |
| Figure 56: Distribution of passengers (sums) per day of week. ....   | 108 |

---

|  |     |
|--|-----|
| Figure 57: Retail sales for each day within research period. ....                            | 113 |
| Figure 58: Accumulated sales per country (ABC-curve, summer season). ....                    | 114 |
| Figure 59: Accumulated sales per country (ABC-curve, winter season). ....                    | 114 |
| Figure 60: Top 15 countries in sales (summer season). ....                                   | 114 |
| Figure 61: Top 15 countries in sales (winter season). ....                                   | 115 |
| Figure 62: Distribution of retail sales (sums) per retail area (basis: actual data). ....    | 116 |
| Figure 63: Distribution of retail sales (sums) per day of week (basis: actual data). ....    | 116 |
| Figure 64: Sales per departing PAX, based on daily average. ....                             | 117 |
| Figure 65: Sales per departing flight, based on daily average. ....                          | 118 |
| Figure 66: Distribution of passengers (sums) per retail area (seasonal plan). ....           | 119 |
| Figure 67: Distribution of passengers (sums) per day of week (seasonal plan). ....           | 119 |
| Figure 68: Gate changes resulting in changes of retail area. ....                            | 124 |
| Figure 69: Excerpt of a sample gate allocation as procuded by the simulation workbench. .... | 135 |
| Figure 70: Comparison of actual figures to those of the baseline scenario. ....              | 137 |
| Figure 71: Application of methodology at other airports. ....                                | 152 |
| Figure 72: Screenshot of software's function/procedure list. Source: Author. ....            | 173 |

---

|  |     |
|--|-----|
| Table 1: Enablers and inhibitors to alignment. Adopted from Luftman et al. (1999).....                 | 14  |
| Table 2: Selection of literature in the field of business alignment of IT. ....                        | 16  |
| Table 3: Basic definition of retail areas.....   | 62  |
| Table 4: Retail-worthiness of countries (selection) at Frankfurt Airport in 2007.....                  | 65  |
| Table 5: Number of potential combinations in (initially) reduced solution space.....                   | 74  |
| Table 6: Directory to source code of algorithm.....  | 82  |
| Table 7: Combinatorial example (1): reduced solution space. ....                                       | 83  |
| Table 8: Combinatorial example (2): retail area combinations. ....                                     | 83  |
| Table 9: Combinatorial example (3): resulting retail sales.....  | 84  |
| Table 10: Combinatorial example (4): optimum combination. ....   | 84  |
| Table 11: Pseudo-code for different implementations of the factorial function. ....                    | 85  |
| Table 12: Distribution of departing passengers across retail areas for each day of a week [...]. ....  | 108 |
| Table 13: Distribution of (daily average) number of flights per weekday.....                           | 109 |
| Table 14: Retail area factors derived from actual data. ....   | 110 |
| Table 15: Distribution of retail sales (in EUR) across retail areas for each day of a week [...]. .... | 115 |
| Table 16: Sales (in EUR) per departing passenger per retail area and day of week (both seasons). ....  | 118 |
| Table 17: Distribution of departing passengers (seasonal plan) across retail areas [...]. ....         | 119 |
| Table 18: Frequency of occurrences of delay minutes (both seasons). ....                               | 121 |
| Table 19: Frequency of occurrences of standard ground time entries (summer season). ....               | 122 |
| Table 20: Frequency of occurrences of standard ground time entries (winter season).....                | 122 |
| Table 21: Frequency of occurrences of standard ground time entries (both seasons). ....                | 123 |
| Table 22: Correlation coefficients for daily aggregated (averaged) [...] .....                         | 125 |
| Table 23: Correlation coefficients for passengers, sales [...].....                                    | 126 |
| Table 24: Definition of baseline scenario.....   | 130 |
| Table 25: Scenario definitions (Group 1). ....   | 132 |
| Table 26: Scenario definitions (Group 2). ....   | 133 |
| Table 27: Scenario results (Group 1), sorted by sales result. ....                                     | 138 |
| Table 28: Violations of alliance rule per day of week. ....  | 139 |
| Table 29: Violations of alliance rule: frequency classes. ....   | 140 |
| Table 30: Scenario results (Group 2), sorted by sales result.....                                      | 141 |
| Table 31: Scenario results (Group 3).....  | 143 |

As the creation of this thesis would not have led to such a satisfying result without the involvement of many people, this is to those who deserve special thanks.

In particular, I would like to thank my supervisor Dr Romano Pagliari. Besides his valuable expertise, his confidence and patience throughout the past five years very much contributed towards achievement of this work on a part-time basis.

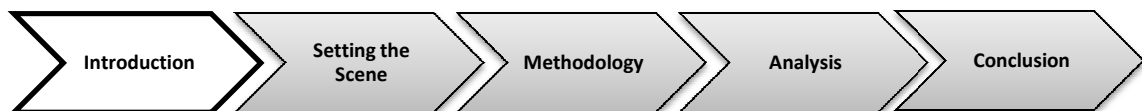
Many colleagues at Fraport, friends and loved ones are thanked for continuous support.



---

|        |   |
|--------|---|
| 2-D    | Two-dimensional                         |
| A/L    | Airline                                 |
| A/P    | Airport                                 |
| ACI    | Airports Council International          |
| ADM    | Airport Data Management                 |
| ASI    | Airport Systems Integration             |
| ATC    | Air Traffic Control                     |
| ATD    | Actual Time of Departure                |
| BHS    | Baggage Handling System                 |
| BPR    | Business Process Re-engineering         |
| CDM    | Collaborative Decision Making           |
| COM    | Component Object Model                  |
| DDE    | Dynamic Data Exchange                   |
| DF     | Duty free                               |
| DLL    | Dynamic Link Library                    |
| EPOS   | Electronic Point of Sale                |
| ETD    | Estimated Time of Departure             |
| EU     | European Union                          |
| F&B    | Food and beverage                       |
| FIDS   | Flight Information Display System       |
| FK     | Foreign Key                             |
| FSS    | Flight Scheduling System                |
| GA     | Genetic Algorithm                       |
| GAP    | Gate Assignment Problem                 |
| GHS    | Ground Handling System                  |
| GUI    | Graphical User Interface                |
| IATA   | International Air Transport Association |
| ICAM   | Integrated Computer-Aided Manufacturing |
| IDE    | Integrated Development Environment      |
| IDEF0  | Integrated Definition 0                 |
| IDEF1X | Integrated Definition 1 extended        |
| IP     | Integer Problem                         |
| LP     | Linear Programming                      |
| NP     | Non-deterministic Polynomial Time       |
| OAG    | Official Airline Guide                  |
| OLE    | Object Linking and Embedding            |
| PAX    | Passenger(s)                            |
| PC     | Personal Computer                       |
| PNR    | Passenger Name Record                   |
| PSS    | Personnel Scheduling System             |
| QAP    | Quadratic Assignment Problem            |
| QoS    | Quality of Service                      |
| RAM    | Random Access Memory                    |
| RAS    | Resource Availability System            |
| RMS    | Resource Management System              |
| SDR    | Special drawing rights                  |
| SPT    | Simplifying Passenger Travel            |
| SQL    | Structured Query Language               |
| STD    | Scheduled Time of Departure             |
| U.K.   | United Kingdom                          |
| U.S.   | United States (of America)              |
| VBA    | Visual Basic for Applications           |

# *1. INTRODUCTION*



## 1. INTRODUCTION

This thesis explores the complex relationships between airport operations, retail management and information technology. It focuses attentions to the commercial imperative associated with retail activity and the operational requirements of efficient airport operations. Although many quantitative aspects included, this thesis does not claim to be an operations research work. However, it aims to support management with findings based on both qualitative and quantitative research.

### 1.1 General

The overall purpose of this chapter is to provide the reader with a general understanding of the research project. Therefore, an outline comprising the aim and possible objectives is given along with the topic's placement within the air transport industry's context.

Past research focused on the relationship between investment into information technology (IT) and the outcome on the (financial) business side. Usually, positive correlation is found, but causal relationship unclear (Hu and Quan, 2005). This research project aims to overcome the most common deficit identified in previous pieces of research: Lack of detail and causality dilution.<sup>1</sup>

*From an IT perspective:* As with any resource (e.g. human labour, land or capital) it is not the pure existence of IT as such that contributes to a firm's success, rather than its intelligent use. Contribution can be in form of commodity services (e.g. email, desktop office support) or in form of highly specialized support for a variety of functions within an enterprise. However, there is no 'one is more important than the other'. Important is the balance of the two. Furthermore, the information intensity of a business process is proportional to the contribution of IT towards a good process performance (compared to less information intensive processes).<sup>2</sup>

*From an airport's perspective:* As part of a highly regulated industry, airports are left little space to increase their revenues. However, the retailing business has proven to contribute to increasing revenues for airport operators.<sup>3</sup> Consequently, airport management should focus on its retail segment. Nevertheless, it needs to be taken into

---

<sup>1</sup> Financial performance measures are usually subject to a variety of influences other than information technology. So, it is often difficult to separate IT's contribution.

<sup>2</sup> See Porter and Millar (1985).

<sup>3</sup> Cerovic (1998), Freathy and O'Connell (1998 various), Airport Council International (2007) in Graham (2008).

account that smooth airport operations are a precondition for successful retailing business at an airport.

*From both perspectives:* To guide passengers physically through retail offering strongly depends on operational processes, one of them being stand and gate allocation. In complex, information intensive environments, the gate allocation process is usually supported by specialized IT systems.<sup>4</sup>

This is exactly where this paper steps in to evaluate information technology's contribution towards retail success at an airport.

## **1.2 Motivation**

The author has chosen this topic for two reasons.

With an airport operations background in mind, it is seen as a challenge to 'think out-of-the-box' in terms of combining operational and commercial airport processes. In discussions he has often found that operational reasons prohibit potential improvements regarding the overall airport product. Unfortunately, without any deeper investigations, possible variations of the way business is conducted have not been pursued. It is the objective to demonstrate that varying processes or business rules enables an improvement of the overall business outcome.

Secondly, from his past role as Corporate Information Architect the author is challenged to demonstrate the potential to be found within intelligent usage of existing information.<sup>5</sup> In context with the aforementioned, this implies the merge of information from different business processes.

In a more specific way the thesis title paraphrases the outlined research endeavour.

---

<sup>4</sup> A more detailed view on 'information intensity' (Porter and Millar, 1985) is provided in Chapter 3.2.1.

<sup>5</sup> The word 'information' is used as a general term, whereas the word 'data' refers to information in a digitized form (e.g. in an electronic data base or in electronic files).

### 1.3 Thesis title

For the purpose of this paper the main elements of the thesis title are to be understood as described below.

#### *Role of information technology*

The role of information technology is often described around terms like ‘strategic alignment of IT’ or simply ‘commodity’. Within this paper this considerable bandwidth is reduced to business process support by means of information distribution and applied methods of operations research (e.g. simulations).

#### *Airport Business*

The main players in focus of research are the airport operator, airlines and retail store operators (includes duty free, speciality retail, food & beverage). Nevertheless, other partners in the airport community like air traffic control, immigration/border control, customs and ground handling service providers also contribute to the airport business environment and would need to be considered regarding any proposed changes. However, the research topic targets at a single airport per investigation with the main players in mind as mentioned above.

#### *Resource Management*

In general, airport resources include airport slots, runways, taxiways, parking stands, gates, terminal space, check-in counters, baggage belts, and facilities for customs or border control. It has to be taken into account that planning the use or allocation of a resource usually is accompanied by the allocation of adequate personnel.

In this paper the process referred to comprises the resources of gates. A ‘stand’ is synonymous with the parking space that an aircraft needs, in accordance with specific requirements (e.g. aircraft size versus stand size). A ‘gate’ is synonymous with the space within a terminal building from where the boarding process starts. Boarding can either start into an aircraft (contact gate) or into a bus taking the passengers to an aircraft parked on a remote stand (bus gate<sup>6</sup>). Research undertaken assumes that in case a gate could be allocated, there would be a contact stand associated with this gate, or a remote stand would have to be allocated for that flight. Staff dispatch or integrated allocation planning for other types of resources are not subject to the research project.

---

<sup>6</sup> This includes a setup where passengers leave the terminal building via stairs and walk across the apron area towards the parking aircraft they depart with.

### ***Retail-Weighted Approach***

As the success of airport retailing depends on many factors<sup>7</sup>, possible support is to be found in different areas. The research project addresses one of the most influential factors towards retail spending: the nationality of the (departing) passenger.

‘Retail-weighted’ in the context of this paper means to harmonize the distribution of passengers with suitable nationality through allocation of flights (aircraft) to favourable retail areas at an airport. Nonetheless, a defined minimum of operational requirements will need to be considered within the gate allocation process.

### ***Airports***

The main characteristic of airports in focus of research is that their gate resources are constrained in a way that allocation cannot be done without planning. This implies that there is strong demand for gates (departure and arrival) and the retail offering is not the same for each gate. Airports with a centralized retail zone are in a position to offer service provided by the same retail stores to all passengers of all flights. In such a constellation the gate allocation does not need to be improved<sup>8</sup> towards retailing, because all passengers would use the same retail area. However, in case of a more heterogeneous offering (e.g. decentralized stores, closer to the gates, different stores, different retail areas) it might be worth it matching the retail offer with passenger demand.

So, capacity-constrained does not mean scarce resources in terms of the runway system or terminal space in general, but focuses on the inter-dependency of retail offering and available gates at a specific point in time.

Having outlined the main elements of the thesis title, the next section of this chapter specifies the project’s aim and objectives.

## **1.4 Aim and objective (demarcation)**

The following defines more precisely what is aimed to achieve (and what not). Thereby it will be possible to evaluate later on, whether the steps chosen in methodology have led to successful results.

---

<sup>7</sup> Freathy and O’Connell (1998c, 2000a), Omar (2001), Kim and Shin (2001), Geuens et al. (2004).

<sup>8</sup> It has to be mentioned that in this paper the word forms of ‘improvement’ are given preference to the word forms of ‘optimization’. This aims to avoid any ambiguity with the meaning of ‘optimization’ in the field of operations research. The latter is not subject to this thesis.

The *overall aim* of the research project is formulated as follows:

To appraise in how far the use of information technology may foster a (AIM) core airport operational process (i.e. gate allocation) taking into account the requirements of an airport commercial process (i.e. airport retailing).

However, as the overall aim is still broad, it has been divided into *single objectives* to support its achievement. The objectives are:

To determine the limits of general purpose methods to model business (O.1) alignment of IT, and placement of the research topic within existing frameworks.

To develop a possible extension or a more specific application of existing (O.2) methods in a way that the derived methodology may be used for further processes in the airport business environment.

To construct a conceptual model describing the relationship between (O.3) airport retailing and gate allocation. The model should be based on process structure (in form of a business process decomposition), as well as on information structure (in form of a data model). Furthermore, it should be quantifiable for later simulation purposes.

To develop an algorithm for the gate allocation process that copes for the (O.4) needs of supporting the retailing process (i.e. an increased sales result).

To develop an independent simulation environment with implementation (O.5) of the algorithm as outlined in O.4.

---

Upon fulfilment of the above objectives, it should be possible to verify the following *statement*:<sup>9</sup>

Retail revenue (sales)<sup>10</sup> at capacity-constrained airports<sup>11</sup> can be increased (*H.1*) by applying specific criteria (rules, preconditions) in the gate allocation process.

The following section briefly outlines the areas where the research project aims to contribute to the body of knowledge.

### **1.5 Potential contribution to knowledge**

Prior to a more detailed discussion in the literature review, this section provides the reader with the main areas aiming to contribute towards the body of knowledge.

Following the methodology as outlined in Porter and Millar (1985), the concept of information intensity has been partially applied to the airport business for the first time.

Furthermore, in a generic model two specific airport processes have been combined in a way to show a potential increase in their output. As a major part of this, a retail sales improving multi-objective gate allocation algorithm has been developed for implementation on standard PC hardware. In consequence, this generic model<sup>12</sup> has been implemented in form of a simulation workbench (piece of software), so that it has been possible to carry out sensitivity analyses. No other case study applies a similar large amount of real airport data to a gate allocation algorithm, and no research has been found with the above improvement objective.

Finally, (based on real world figures) new insight aims to be provided into the financial potential that is incorporated in combining the two airport processes examined when supported by a specific IT solution.

---

<sup>9</sup> Due to the fact that this thesis does not aim to be of quantitative operations research nature, the word 'statement' is given preference to the word 'hypothesis'.

<sup>10</sup> Depending on the retail business model of an airport, sales may be revenue of the airport operator or of the retail operator. In case of the latter, only a proportion of sales will be revenue for the airport. In the context of this paper it is aimed to increase overall sales, regardless how it will be shared amongst the business partners.

<sup>11</sup> To be understood as described in explanation of thesis title.

<sup>12</sup> The quantified conceptual research model.



## 1.6 Organisation of this paper

**Chapter 2** sets the scene. After a brief description of the current status of the airport industry, a literature review provides more insight regarding current discussions in the areas of ‘business alignment of IT’, ‘gate allocation’ and ‘airport retailing’. Deriving from that, the research question is formulated concluding the chapter.

**Chapter 3** presents the research approach in form of the methodology. This comprises a detailed view on the chosen business processes, the formulation of the conceptual research model, the data needed, and a quantification of the model. As a focal point of the research project, a sales improving gate allocation algorithm is developed along with an explanation for its intended application in form of scenarios. Finally, and within this piece of research a major portion of work, the self-developed simulation environment, is introduced in more detail. In order to derive insight out of the simulation runs, within **Chapter 4** different scenarios are discussed, and the results of simulation runs are analysed for sensitivity. Having gained an impression about operational and financial feasibility of certain set-ups, **Chapter 5** discusses the approaches’ transferability to other airports or players in the airport business and proposes actions to transfer the findings into airport policy. Limitations of the research undertaken are outlined as well, and finally, areas for future research are proposed and conclude this thesis.

## *2. SETTING THE SCENE*



## 2. SETTING THE SCENE

The research topic's comprehensive nature leads to the situation that there is no field in literature comprising all sub-topics. So the objective of this chapter is to provide a basis for the research project and to familiarize the reader with the individual topics. Before the fields of information technology alignment towards business, airport gate allocation and airport retailing will be discussed, an introductory chapter aims to outline the key areas an airport may concentrate on to improve its competitive situation.

### 2.1 Success factors in the airport industry

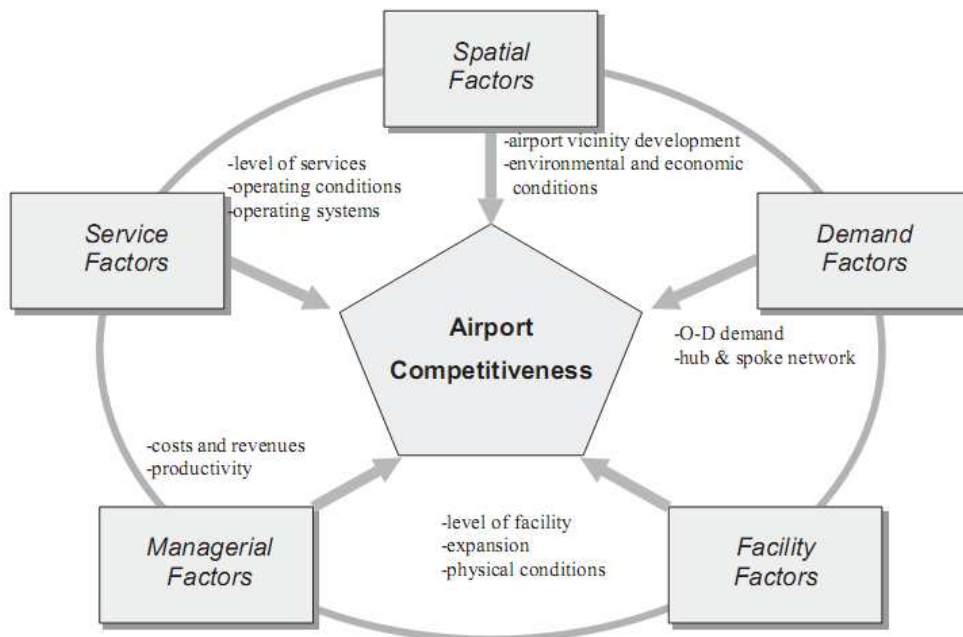
The airport business can be seen both being a service industry and being part of (national) infrastructure. On the one hand, demand for service may change at short notice and also regarding its characteristics. This usually involves changing streams of revenue from the customers (e.g. airlines, passengers). On the other hand, the infrastructural characteristics of this industry incur capital cost for long periods on the owners of an airport (often up to thirty years, e.g. for runways). So, a major challenge within airport management would be to address the different business cycles found in this industry (i.e. amongst the different business partners) with proper investments.

In a more general way, Michael Porter's widely known concept of the 'competitive forces' (Porter, M.E. 1980 and 1998) places any company into its competitive environment, determined by the power of buyers, the power of suppliers, the substitute of products and services, possible new entrants and existing competitors. The same applies to airports (e.g. high speed rail as a substitute for certain air services).

Based on Porter's works, Park (2003) applies the above to the airport industry. He identifies factors to structure an airport's competitive advantage (see Figure 1). In the context of the research topic a core question focuses around the potential contribution of information technology (IT) within these groups of factors.<sup>13</sup> Many of the factors are predetermined e.g. by the location of an airport. In such cases technology may compensate for disadvantages (e.g. specialized radar equipment to compensate for prevailing conditions of poor visibility due to bad weather). More examples demonstrate that IT may support various areas in the airport business.

---

<sup>13</sup> Examples for IT support may be for each of Park's factor groups: *Spatial factors*: noise monitoring; econometric airport model on spreadsheet. *Demand factors*: route development tools; market research tools; *Facility factors*: computer aided facility management; capacity simulation. *Managerial factors*: management information systems; business intelligence applications. *Service factors*: passenger way finding support; dispatch of cleaning staff.



**Figure 1: Factors of airport competitiveness. Source: Park (2003, p. 354).**

Usually, there are certain categories, which IT support may be divided into:

- control of processes or machines (e.g. baggage sortation)
- information distribution (e.g. display of flight information to public)
- administrative support (e.g. invoicing, automated purchase orders, decision support, planning, marketing).

In addition, there are different technical elements within IT that contribute towards a successful IT function, like:

- hardware infrastructure (e.g. physical networks, storage units, processing units)
- software infrastructure (e.g. operating systems, message bus, mail system)
- commodity software applications (office support like spreadsheet, word processing, electronic calendar)
- specialized software applications (e.g. flight information display system, baggage tracking system, airport operational database).

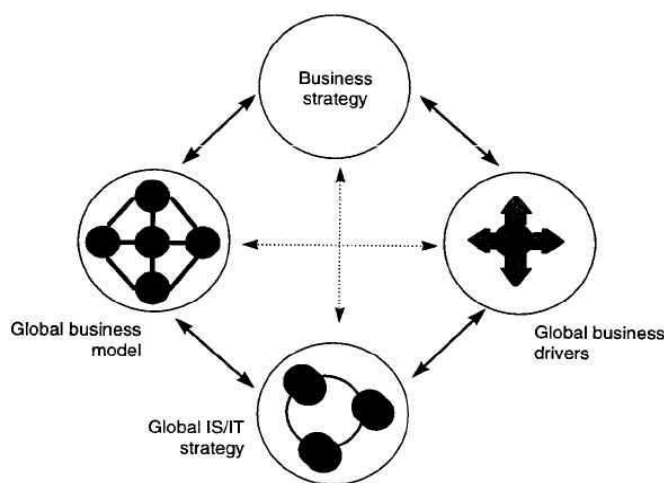
Besides these rather tangible elements there is also the intangible element of managing the IT function (usually referred to as IT management or IT governance). It is assumed that only if all different aspects are carefully aligned, IT may contribute towards the success of an airport. Furthermore, the different players in the airport context may contribute to a variety of the above aspects. An airline, for example, may be provider of

IT infrastructure at U.S. American airports, but paying to use the same type IT infrastructure at European airports. Examples for the above application of IT are subject to various works.<sup>14</sup> To obtain more insight into a potential contribution of the IT role at an airport the topic is further explored in the following chapters.

## 2.2 Alignment of business and information technology

Very well-covered in literature, 'alignment' is usually thought of a business strategy's match with its support through the IT function within an enterprise.<sup>15</sup>

Although the airport business is a world-spanning industry, global alignment models like that of Peppard (1998), apply to airports not in every aspect. In his model (see Figure 2) the global business drivers definitely would also need to be considered in airport business. Nevertheless, a global business model as referred to in his framework would assume an airport to be an international player. But only some of the major airport operators or large construction companies are. However, applied to the research context usually the airlines incorporate elements of a global business model (e.g. in form of alliances or stations at a variety of international destinations they serve).



**Figure 2: Strategic alignment framework (for a global enterprise).**  
Source: Peppard (1998, p. 10).

Applied to the IT strategy context it means to provide the IT function in alignment with the business requirements at any location it is needed. In consequence, an international airport (not being part of a global airport group) would need to address the (global) needs of its airline partners on the airport premises.

<sup>14</sup> Bloem, E.A., Blom, H.A.P. and Schaik, van F.J. (2002); Bonnke, J. (1999); Button, K., Lall, S., Stough, R. and Trice, M. (1999); Esper, T.L. and Williams, L.R. (2003); Feldman, J.M. (1999); Forster, P.W. and Regan, A.C. (2001); Hill, L. (2002); Montealegre, R. (2000); Neufville, de R. (1994); Pitt, M., Wai, F.K. and Teck, P.C. (2002); Reinheimer, S. (1998); Wiese, P. (2003 a, b); Seamster, Th.L. and Kanki, B.G. (2002). (The latter more in the context of an airline's electronic flight bag.)

<sup>15</sup> A complete overview of the 'alignment' – topic would be far beyond the scope of this paper. Nevertheless, models and concept relevant for this work are introduced. A more comprehensive review can be found in Pollalis (2003).

For example, this can be to simply comply with the various customers’ IT standards or the other way round to support them using local but airport-wide technology standards. Without emphasis on a global aspect (but with possibility to consider it) Henderson and Venkatraman (1993) also point out that the business function and the IT function would need to be linked together (Figure 3). In addition to this functional fit, they claim that within each function there has to exist a strategic fit. This requires both business strategy and IT strategy to organize each their own support organization (administrative infrastructure, IT architectures, processes, skills).

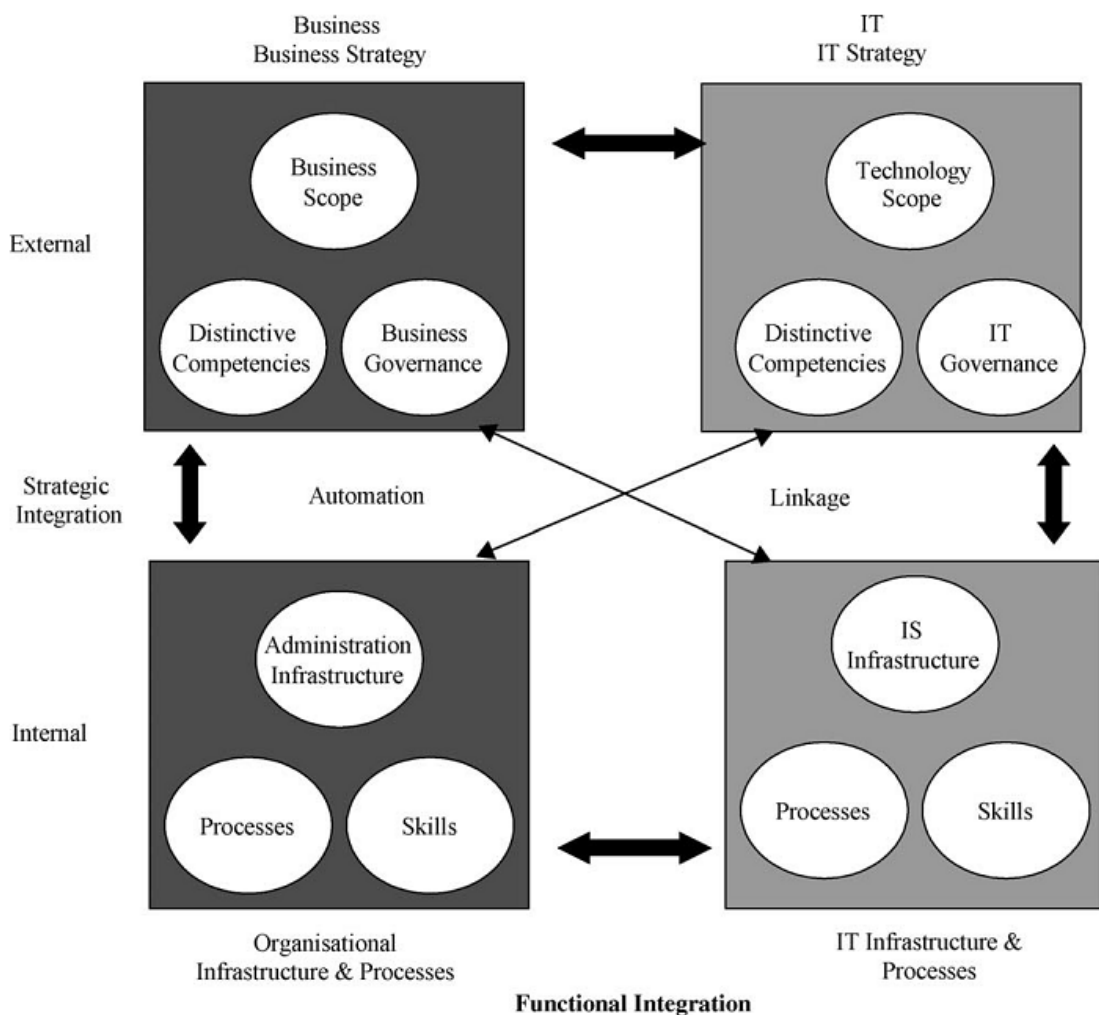


Figure 3: Strategic alignment according to Henderson and Venkatraman. Source: Avison, D. et al. (2004).

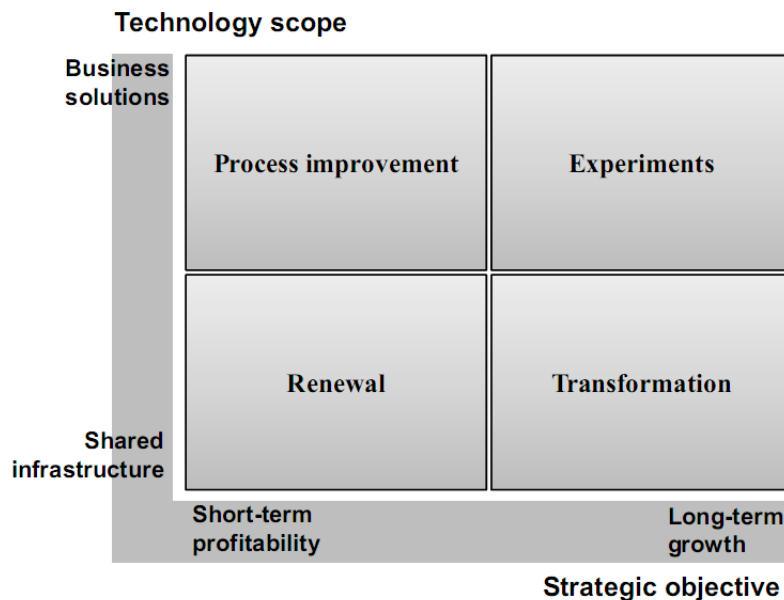
Not extending existing models, but asking for the reasons of possible alignment Luftman et al. (1999) focussed on the enablers and inhibitors in their study<sup>16</sup>. They found that there are the following contributing factors regarding alignment:

<sup>16</sup> 1992-1997, more than 500 fortune 1000 U.S. firms in 15 industries.

| Enablers   | Inhibitors   |
|--|--|
| <ul style="list-style-type: none"> <li>• Senior executive support for IT</li> <li>• IT involved in strategy development</li> <li>• IT understands the business</li> <li>• Business/IT partnership</li> <li>• Well-prioritized IT projects</li> <li>• IT demonstrates leadership</li> </ul> | <ul style="list-style-type: none"> <li>• IT/business lack close relationship</li> <li>• IT does not prioritize well</li> <li>• IT fails to meet its commitments</li> <li>• IT does not understand business</li> <li>• Senior executives do not support IT</li> <li>• IT management lacks leadership</li> </ul> |

**Table 1: Enablers and inhibitors to alignment. Adopted from Luftman et al. (1999)**

Most of the alignment models assume that there is a certain budget that the IT function spends in order to support the business function in a desired way. In Table 1 arguments like ‘IT understands the business’ or ‘well-prioritized IT projects’ suggest such an assumption. However, the IT budget needs to be split into different portions in order to deliver IT services in the long run. For example, it may become necessary to invest into IT infrastructure in order to cope with future business requirements (compare Figure 3). But in case business strategy changes in a way that its innovative component shall be stressed, a different type of IT investment would become necessary. Although in the models above not explicitly mentioned, the investment function needs to be a core part of business/IT alignment, too. A framework that supports this idea is presented by Ross and Beath (2002).



**Figure 4: An IT investment framework. Adopted from Ross and Beath (2002).**

Such an approach may help to decide which projects to realize e.g. within an annual (or rolling) investment process. For example with the help of a scoring model, the projects can be prioritized against each other, but only within the category they belong to. The

categories (process improvement, renewal, transformation, experiments) themselves have to be prioritized (according to business requirements and consequent IT necessities) and then budgeted.

Given the situation that projects have been approved by internal bodies of an enterprise there should be a clear understanding of the output not only of the project itself, but moreover of the investment throughout its entire life (or at least to an agreed point in time, e.g. some point after amortization). Therefore, it is necessary to define and measure the desirable outcome, which justified the investment upfront. A well-known approach to manage this from different perspectives has been presented by Kaplan and Norton (1992). As can be observed in Figure 5 there are no explicit links towards the IT function. Nevertheless, it complements Porter’s (1980) model, and Park’s (2003) ideas also fit into the idea of the balanced scorecard. For example, the scorecard’s ‘customer perspective’ may fall under Porter’s ‘power of buyers’ and Park’s ‘service factors’.

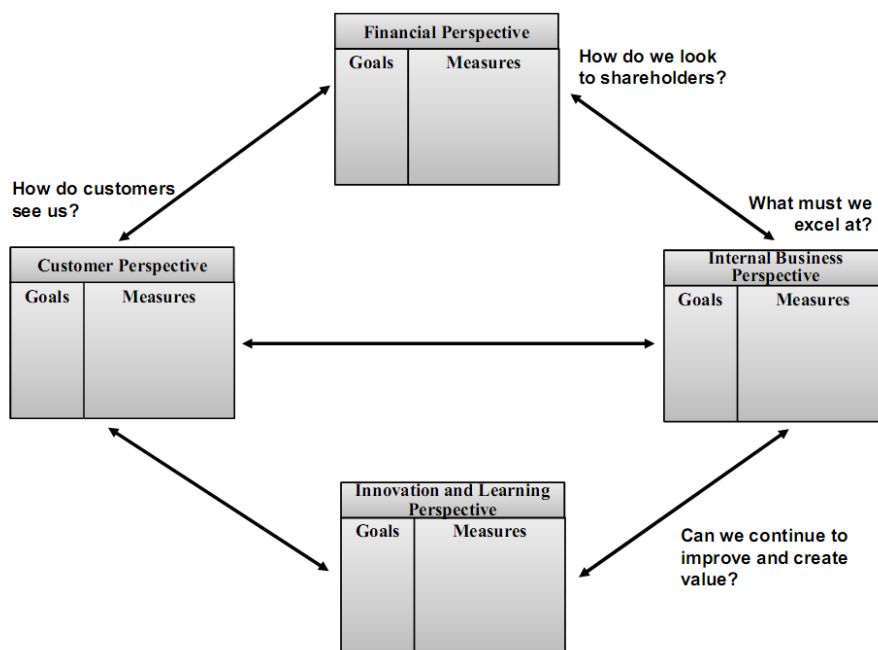


Figure 5: Structure of balanced scorecard. Source: Kaplan and Norton (1992, p.72).

In the scorecard the respective goal may be to increase customer satisfaction, measured in a monthly index, based on feedback from customers.

So far, models have been identified that describe the relationship between the IT function and the business function. Furthermore, it is possible to describe the output that shall be generated through investments.



A number of additional works address the aforementioned alignment question under different aspects and partly incorporate the performance aspect.

A summary is provided in Table 2.

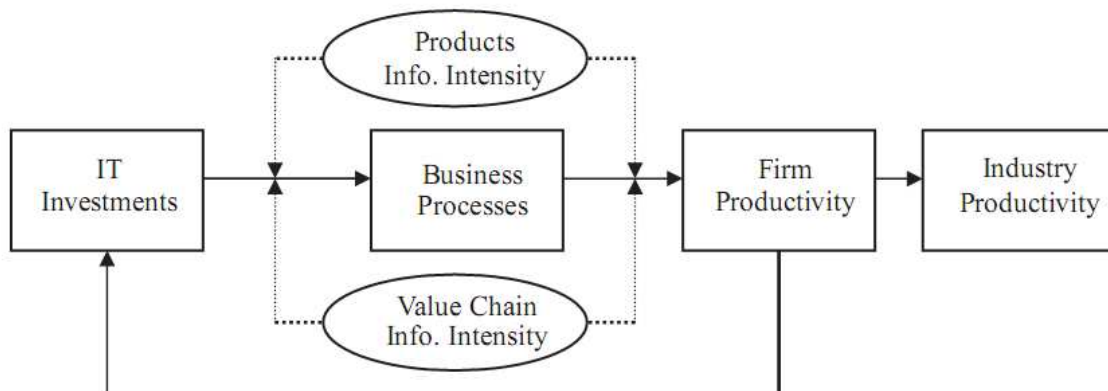
| Main Aspect  | Relevance  | Source  |
|--|--|---|
| Case study   | Comprehensive application of alignment issues  | Feurer, Chaharbaghi, Weber and Wargin (2000)      |
| Case study   | Relevance of telecommunications in different industries as input for IT strategy   | Browdy (1999)                                     |
| Case study   | Association of IT investment and survival of a company (in the U.S. railroad industry).  | Cline and Guynes (2004)                           |
| Case study   | IT alignment planning process  | Peak et al. (2005)                                |
| Elements of alignment  | Foundation works on alignment  | Luftman et al. (1993), Luftman (2003)             |
| Challenges of alignment  | Organisational conditions, economic return of IT investments   | D'Souza and Mukherjee (2004).                     |
| Convergence of IT and business process management                                  | IT as an enabler for business performance management   | Melchert and Winter (2004)                        |
| Interaction of IT and business unit  | Dependency of internal IS functions on interaction model   | Gordon and Gordon (2000)                          |
| IT financial management, service management  | 3-tier IT service hierarchy  | Cohn (2003)                                       |
| Marketing  | The market factor influenced by IT strategy.   | Adams, Haines and McLellan (2003)                 |
| Performance measurement (benchmarking)   | Applying Performance Metrics to internal IS organisation   | Clark / Lee (2002)                                |
| Performance measurement (general)  | Foundation work on performance measuring   | Eccles (1991)                                     |
| Performance measurement (IT)   | Assessment of productivity impact  | Loveman (1994)                                    |
| Performance measurement (IT)   | Measures for: IT effectiveness and IT efficiency   | Rau (2004)  |
| Strategy   | Shows link between business strategy and business performance measuring  | Kaplan and Norton (2004)                          |
| Strategy   | Shows different stakeholder views on strategy  | Kaplan and Norton (2000)                          |
| Strategy   | Factors that influence strategic goals (management-centric view)   | Clarke (1994)                                     |
| Strategy   | Gap between opportunity through IT and its utilisation. Strategic value of IT.   | Benjamin, Rockart, Scott, Morton and Wyman (1984) |
| Strategy, IT applications, IT operations, IT architecture, financial tools, people | Comprehensive approach, taking alignment into account as well as governance (both corporate and IT). Highlights limitations of financial tools usually in place. | Mack and Frey (various 2002, 2003)                |

**Table 2: Selection of literature in the field of business alignment of IT.**

The different aspects within the works above show that the IT function in a company consists of many individual disciplines that need to be taken into account (e.g. operations, architecture, development, finance). Nevertheless, to this point nothing is said in respect to a (quantifiable) contribution of the IT function towards business performance.

In accordance with most models Pollalis (2003) acknowledges the need for integration of the IT function and the business function. His study surveyed business and IT personnel from banking institutes. Applying a “Gestalt approach or *taxonomic perspective* [that] attempts to analyze the various components of organizations simultaneously, without assuming any directionality of relationships”<sup>17</sup> he identified that organizations with a consistent and high level of technological integration, functional integration and strategic integration outperform those with poor technological integration. Functional integration is seen as a prerequisite (but not being sufficient) for a high performing organization.

At industry level Hu and Quan (2005) examined eight different industries regarding the impact of IT investments on productivity. They agree that according to Porter and Millar (1985) those firms would benefit most from investments into IT whose value chain and products are most information-intensive.<sup>18</sup> Figure 6 provides an outline of their research model.



**Figure 6: Impact of IT investments on productivity at industry level. Research model. Source: Hu and Quan (2005, p. 43).**

However, they observed that in previous research the question of causality has often been neglected. So would high productivity be due to a high level of IT investment, or is the latter only possible because of high productivity? Hu and Quan addressed this

<sup>17</sup> Pollalis (2003, p. 476).

<sup>18</sup> Regarding the methodology of Porter and Millar, see also Chapter 3.2.1.

with a Granger causality model that “can simultaneously test all possible causal relationships between two variables without any predetermined causal assumptions.”<sup>19</sup> Their results show that in six out of eight industries IT investments positively contribute towards an industry’s productivity.<sup>20</sup> However, applying the concept of information intensity, they assume that industries with a high ratio of IT investment to total capital investment bear high value chain information intensity. No further causal classification of industries within Porter and Millar’s Information-Intensity-Matrix has been conducted. No industry-specific reasons have been provided that explain the observed causality.<sup>21</sup>

Glazer (1991, p.6) supports that the “notion of information intensity provides an operationalization that can be used as the basis for a series of hypotheses about the effects of the changing information environment on business activity.” He suggests a model that exploits the value of information within transactions.

Hu and Quan note that in “a recent meta analysis of firm-level IT payoff studies, Kohli and Devaraj (2003) also suggested that productivity-based measures are more suitable for capturing IT investment payoff than profitability-based measures because productivity measures are less likely to be confounded by external factors.”<sup>22</sup>

Many of the aforementioned approaches lack to produce (or do not ask for) the details needed in order to be used as a comprehensive tool for management guidance. Gartner’s<sup>23</sup> Total Value of Opportunity (TVO) Approach<sup>24</sup> is found to provide this completeness and level of detail.

It is “a metrics-based approach to measuring business performance based on three important factors: risk, time and the effectiveness of converting projected value into actual business benefit.” (Apfel, 2002). As such it aims to determine the value of an IT-enabled business initiative<sup>25</sup>, to cope for future uncertainty and to consider alignment through organisational diagnostics.

---

<sup>19</sup> Hu and Quan (2005, p. 46).

<sup>20</sup> And a feedback loop has been detected: high productivity also contributes to high IT investment.

<sup>21</sup> Apart from the general causal relationship in the model of Porter and Millar (1985).

<sup>22</sup> Hu and Quan (2005, p. 43).

<sup>23</sup> Gartner is an IT research firm and consultancy.

<sup>24</sup> Apfel (2002). Apfel and Smith (2003).

<sup>25</sup> Such an initiative is usually spoken of as an ‘IT project’. The latter term is often misleading because usually only projects around IT infrastructure are pure IT projects. The remainder are business projects with IT portions included.

---

The key value questions to be answered by this approach are as follows (Apfel and Smith, 2003): “... ”

- What is the initiative?
- How will we measure the business value?
- What does the technology do?
- How much benefit will we receive?
- How much will it cost?
- How do we take into account future uncertainty?
- Is the enterprise positioned to exploit these capabilities? ...”

For each of the questions there is a set of standard tools or best practices that can be applied to provide for a sufficient answer. The aforementioned balanced scorecard approach and its extension to strategy maps (Kaplan and Norton, 2000 and 2004) linked to performance measures would be such a possible best practice methodology.<sup>26</sup> As mentioned before, a scorecard of its own would not reflect the IT portion of an investment.

In order to link business context to IT investments, Gartner identifies five needs that have to be considered to actually derive business value from IT.

Given the TVO approach and consideration of the question that needs to be addressed in order to realise the benefits from investments into IT the following five pillars are seen as a foundation:

- Strategic Alignment
- Risk (assessment)
- Business Process Impact
- Direct Payback
- IT Architecture

Apart from the last pillar (IT architecture), the first four are independent from a business support function. However, they need to be applied to a specific industry in order to make them work (e.g. using industry-specific key performance indicators to measure and compare business process impact). So, it is the IT architecture that enables (or inhibits) the realisation of a projected business benefit. This discipline constitutes a huge lever regarding benefits realisation. IT architecture or mostly spoken of as enterprise architecture is a very broad field and well covered in literature and research. Many frameworks and models have emerged within this discipline. It would be far

---

<sup>26</sup> Corresponding question within TVO: How will we measure the business value?

beyond the scope of this work to discuss them here. However, for the research topic it is helpful to understand the concept of the different architectural layers.

In practice<sup>27</sup> the author used the illustration in Figure 7 to distinguish between the different architectural layers. In order to achieve both functional and technological integration (as found important for alignment by Pollalis, 2003), it is essential to map the various activities (in business processes) to a corresponding technological implementation. However, the technological implementation is usually observed as a ‘black-box’ by users of IT.

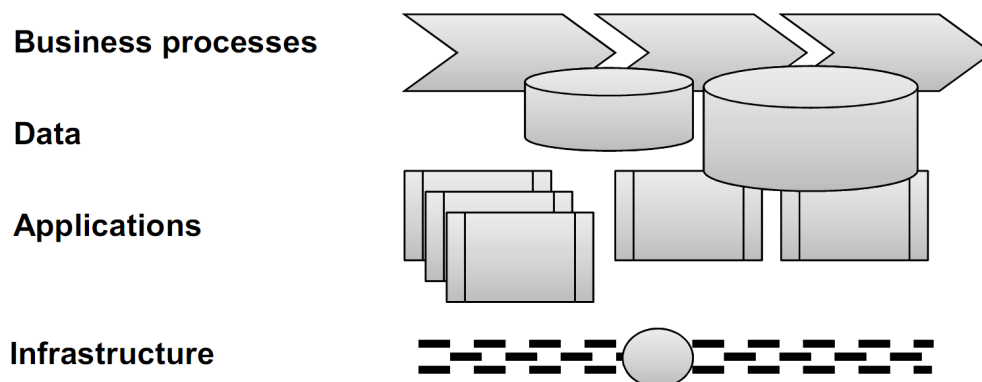


Figure 7: Different architectural layers. (own illustration).

In most cases they perceive IT in form of a software application’s user interface. Usually, via such an interface interactions are triggered and results provided.

In rare cases there is direct access to the data layer of an IT solution. For reasons of data security and integrity such an access is usually handled via the application layer. The infrastructure of an IT solution can be both hardware (e.g. physical network, computers, disk arrays) and software (e.g. operating systems, application servers).

As indicated above, normally there is an interface to the user (‘client’ or ‘frontend’). Furthermore, a ‘backend’ and sometimes a sort of ‘midend’<sup>28</sup> might be part of an IT solution. Each of those ‘-ends’ may be found for each of the architectural layers. For example, a software program may both store and manage data locally within the client-application and exchange data with a centralized data store in the ‘backend’. In such a

<sup>27</sup> Concurrently to research work, the author has designed the future enterprise IT architecture of Frankfurt Airport, covering their business model. Precise verbal communication has been key to successful gathering of business requirements, based on the Rational Unified Process.

<sup>28</sup> This is not exactly the same as ‘middleware’. The term ‘middleware’ is usually applied to messaging software in the ‘midend’.

scenario the software routines that transfer the data between ‘frontend’ and ‘backend’ would be referred to as the ‘midend’ (within the data layer).

Another aspect is important for a proper understanding between IT function and business function, and often looked at in a late stage of requirements gathering: the different life-cycles of an IT solution require different IT environments a solution is embedded into. Usually, a user of IT has in mind the production environment when speaking of e.g. an application. Nevertheless, an IT solution first needs to be developed (or tailored, or configured), and then tested before it is run within the production environment. Furthermore, the management of the production environment as such normally requires an environment of its own: the operations environment (e.g. systems management, network management).

So, the basic four architectural layers are to be discussed in context of the two dimensions (IT environment and the ‘-ends’). Figure 8 summarizes this.

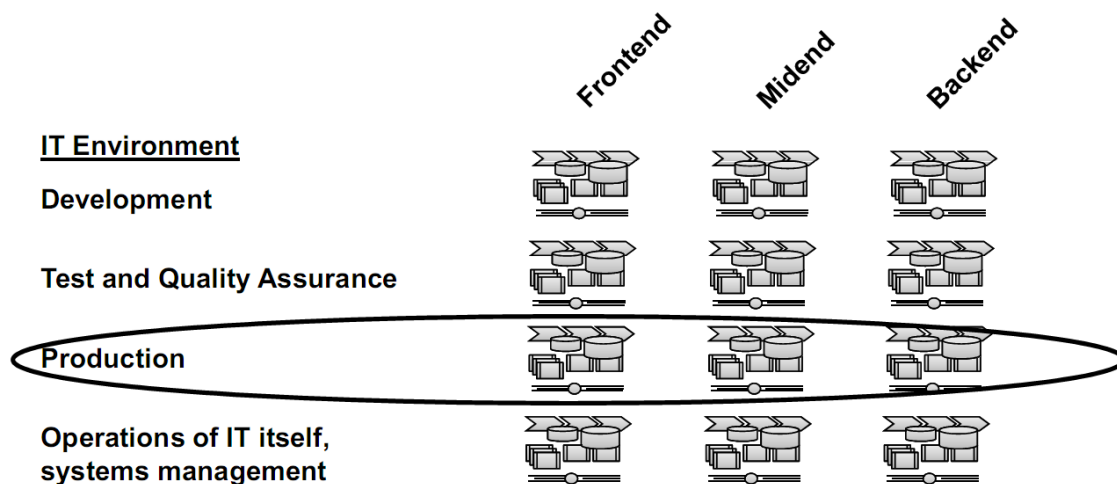


Figure 8: The different IT architectural layers in context of their two dimensions. (own illustration).

This research project seeks IT contribution within the production environment. Nevertheless, such an IT solution should have as few requirements as possible towards the other environments (to keep cost low for development and at run-time).

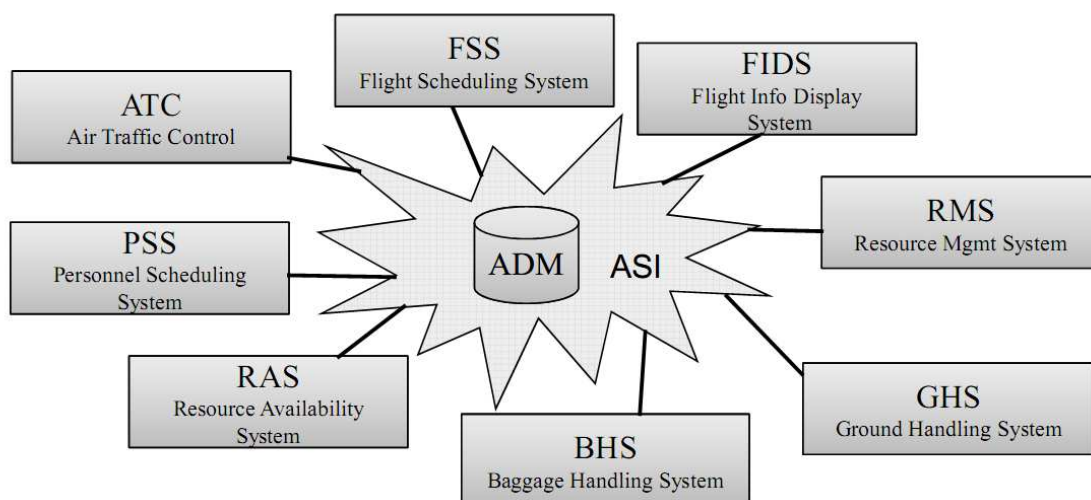
The amount and type of IT solutions found at an airport vary considerably, and are much dependent on size and business model of an airport. The range may be from half-a-dozen IT applications to several hundred IT applications.

As a first common attempt within the aviation industry the International Air Transport Association (IATA) and the Airports Council International (ACI) introduced an implementation scenario for IT solutions at an airport (Figure 9). Although the scenario

is not quite specific regarding the architectural layers and the dimensions, the basic idea becomes evident: A common platform for the shared use of information.

The major IT solutions of the different players at an airport are supposed to use an individual sub-set of information from a centralized airport data store (ADM in Figure 9). At that time, the focus was on current information in respect to flight operations.<sup>29</sup>

A sort of ‘midend’ across all architectural layers is defined by the airport systems integration guidelines (ASI in Figure 9). These basically aim to enable the exchange of information amongst the players (possibly in an automated manner).



**Figure 9: Generic IT systems landscape at an airport according to an implementation scenario by IATA and ACI (April 1998). Illustration adopted by author.**

The implementation scenario as presented in Figure 9 is independent from an airport’s business model as long as the (technical) integration of the business processes (or functions) by means of information exchange is considered.

In respect to alignment of business function and IT function for airports it can be concluded that most of the alignment models may be applied to the research context. However, a lack of causality in the relationship between IT as an input factor and business performance (in form of quantifiable output) limit the statements of many models. Porter’s value chain and the concept of information intensity seem to be a vehicle to identify processes that influence (more than others) business performance in case they were supported by IT.

<sup>29</sup> Nowadays a shared pool of information is also basis for common business intelligence initiatives. Such an initiative may be the shared goal to measure (and analyse) punctuality figures, e.g. in the context of collaborative decision making (CDM).

Thus a model capable to show any (monetary) potential realized upon proper IT support of a business process would contribute as follows:

- existing alignment models may be tested with an actual application
- a methodology around such a model may be applied to explore more business processes in the airport environment – in consequence enabling a holistic view over an airport in that respect.

Most alignment models imply a proper IT support, which basically means a well designed and implemented IT solution. In order to determine the potential that may be realized upon application of such IT support, a specialized IT solution would need to be developed. Its design and underlying information model further contribute to existing knowledge. Referring to objective *O.1* Figure 10 (visually) places the research topic, so far, within existing knowledge and indicates its possible contribution as outlined above.

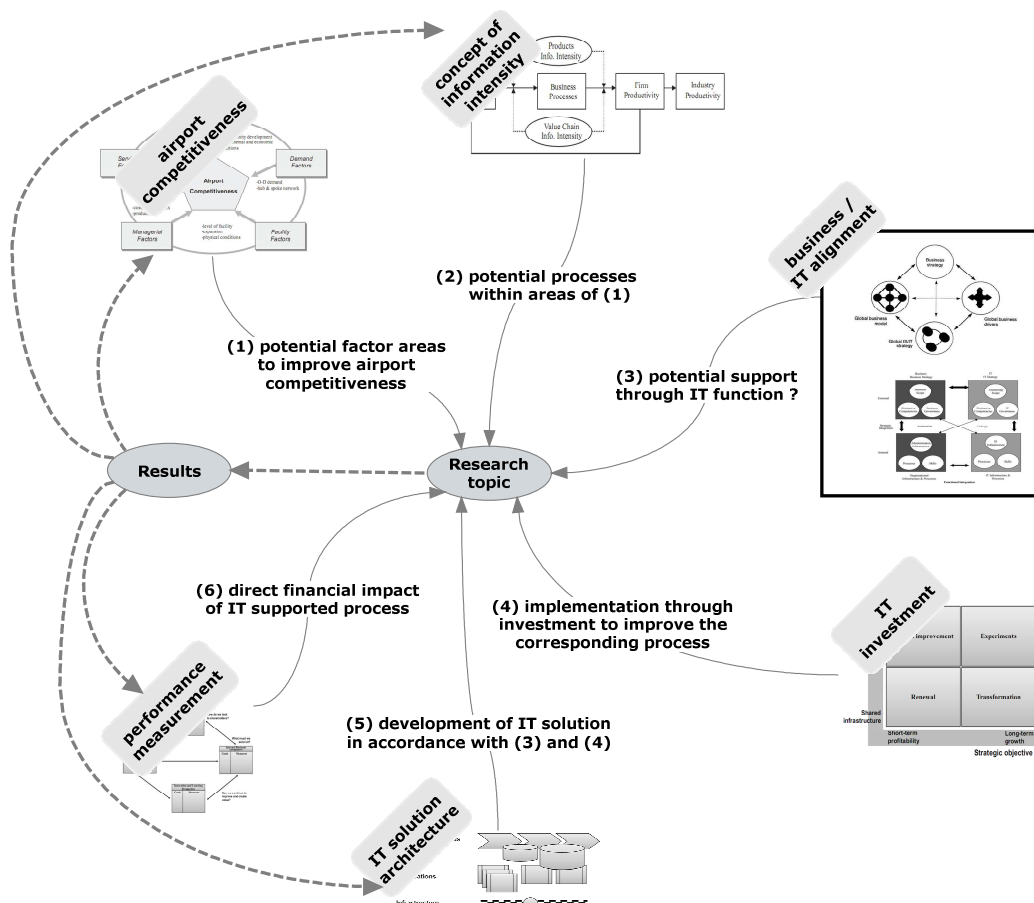


Figure 10: Research topic placed within body of knowledge (of IT alignment).



Furthermore, following the steps (1) through (6) in Figure 10 may suggest a way to investigate additional business processes for potential improvement.<sup>30</sup>

Within the methodology chapters it will be explained why the gate allocation process seems to be a good candidate for IT support. Nevertheless, a brief review of existing research in this field identifies gaps that will further support process selection.<sup>31</sup>

### 2.3 Gate allocation

In literature ‘gate allocation’ is often referred to as ‘gate assignment’. The topic is well known and usually called the ‘gate assignment problem (GAP)’. It is “an easily-understood but difficult to solve problem.” (Haghani and Chen, 1998). Some researchers regard it as an airport task. Some rather see it from an airline perspective. As mentioned in the introductory chapters, this is usually due to the airport model the individual researcher was exposed to while conducting studies. Nevertheless, there are airports where both airport operator and airline(s) perform the gate allocation task either for separate local areas or in a collaborative way for a defined shared local area.

However, as expressed by Murty et al. (2008, p.3), the “dynamic operational environment in modern busy airports, increasing numbers of flights and volumes of traffic, uncertainty (random deviations in data elements like arrival, departure times from flight time tables and schedules), its multi-objective nature, and its combinatorial complexity make the flight-gate allocation a very complicated decision problem both from a theoretical and a practical point of view.”

The GAP is an integer problem (IP), which can be addressed with the linear programming (LP) method. “The basic constraints of the IP are that one aircraft is to be assigned to only one gate and two aircraft cannot be assigned to the same gate when their apron times overlap.” Haghani and Chen (1998, p.440).

Usually, there is at least one objective to solve the GAP for (e.g. passenger walking distance). In such a case the problem type is called a ‘quadratic assignment problem’ (QAP). A detailed discussion is presented in Haghani and Chen (1998) and Dorndorf et al. (2007). “The gate assignment problem is an NP-hard<sup>32</sup> problem (Obata, 1979)” in Haghani and Chen (1998, p. 438). And for quadratic assignment problems that “are NP-

---

<sup>30</sup> In response to objective *O.2* and in preparation for *O.3*.

<sup>31</sup> Because of the complex nature of the topic, the reader is supported to (re-)establish the link between an individual aspect discussed and the overall alignment question in many paragraphs headed ‘[ALIGNMENT ASPECT]’.

<sup>32</sup> NP (non-deterministic polynomial time) is a complexity class in computational complexity theory.

hard (there are no polynomial time bound algorithms for their solution), only implicit enumeration methods are known for solving them optimally.” (Haghani and Chen, 1998, p. 441). Also Lam et al. (2002, p. 104) confirm that “in technical terms, the gate assignment problem is combinatorial in nature, NP-hard, and cannot be optimized easily within a practical time frame.”

This may be one reason for the intense research dedicated to the GAP. As usually, there are computers involved in solving different formulations of the GAP with various methods, also technical advances led to new approaches and implementations.<sup>33</sup> Previous approaches in gate allocation can be clustered by “assignment methods, problem solving methods and objective function used.” (Cheng 1997, p.838).

Assignment methods may be either sequential (following a strict order), parallel (consider all flights and all gates concurrently) or grouped in a problem-oriented way (i.e. combination of the previous two methods). An early implementation of the sequential assignment method is that of Hamzawi (1986). His basic framework (see

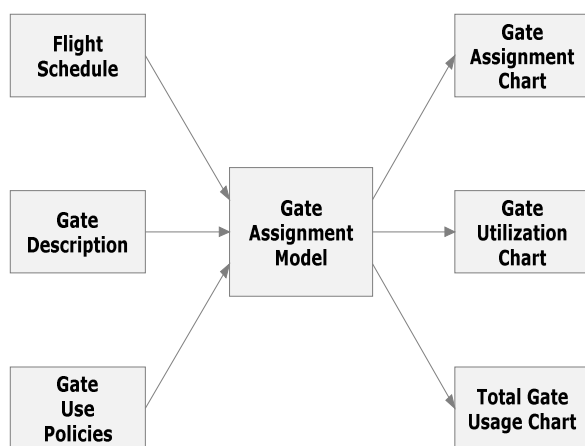


Figure 11) is still valid for most approaches that aim to implement solutions of the GAP. He applied his model’s implementation as a planning tool at several Canadian airports. Also Mangoubi and Mathaisel (1985) followed a sequential approach, but applied a heuristic problem solving method to it.

Figure 11: Model framework according to Hamzawi (1986, p. 193).

However, solutions of higher quality<sup>34</sup> are usually found applying a parallel assignment method. Owing to the combinatorial nature of the problem, Babic et al. (1984) as well as Mangoubi and Mathaisel (1985) found early that by following a pure parallel assignment method, it is barely possible to compute a solution. Hence, the problem-oriented group assignment method promised to work better for many researchers.

Regarding the problem solving method, three groups of approaches are found: mathematical programming approaches, heuristics and knowledge-based approaches.

<sup>33</sup> Compare also the change from mainframe to microcomputers in Hamzawi 1986, p.191.

<sup>34</sup> A solution has a higher quality the more constraints and multiple objectives are considered in solution finding. The closer to reality, the higher is the solution quality in this context.

While the first deliver exact solutions applying various mathematical programming methods like branch-and-bound, linear programming (with relaxation) or pure integer programming, its main disadvantage is the huge amount of computing time necessary.<sup>35</sup> Researchers in this area also claim that it is very complex or impractical to change the objective function, whereas practitioners often regard solutions as too simple, because the objective functions do not reflect real world. Tosic (1992) also stresses the challenges with a large number of flights and gates.

To overcome the timing problem, many heuristics have been tried as problem solving method for all assignment methods. Early approaches have been conducted by Mangoubi and Mathaisel, or by Hamzawi for sequential assignment, and then later for problem-oriented group assignments. According to Cheng (1997) six heuristics have been tested by Zhang et al. (1994) producing favourable results. Gu and Chung (1999) applied a genetic algorithm heuristic to the GAP, and most recently Hu and Di Paolo (2007). Ding et al. (2004) used a greedy algorithm with tabu search. A year later they [Ding et al. (2005)] applied a pure simulated annealing approach and to improve timing, then a hybrid simulated annealing with tabu search approach delivered good results. With a different focus, also Yan and Tang (2007) implemented their model with a heuristic method. However, a drawback of heuristics is often that they are specialized for a certain situation. So, for a different setting of the problem it might become necessary to implement another heuristic.

The third group of problem solving method – knowledge-based approaches – is usually implemented in form of expert systems (Brazile and Swigger, 1988; Shifrin, 1988; Gosling 1990; Srihari and Muthukrisnan, 1991; Su and Srihari, 1993; Cheng, 1997; Cheng, 1998). In this context an expert system is usually understood as a system<sup>36</sup> that comprises the knowledge of personnel, which usually performs the gate allocation task. Such knowledge is captured in form of rules<sup>37</sup>. Furthermore, it may be decided whether a system is used purely for planning purpose or also for real-time assignments. In terms of interaction it may further be distinguished between user-interference during flight operations being permitted, or an autonomously running system. Expert systems are

---

<sup>35</sup> This is due to the combinatorial explosion.

<sup>36</sup> Here a system is the implementation of an assignment method (sequential, parallel, problem-oriented) following the knowledge-based approach as the problem solving method.

<sup>37</sup> There may be different types of rules: [a] hard rules (conflicts); [b] soft rules: [b.1] dependency rules (e.g. between resources), [b.2] preference rules (e.g. airline wishes) [b.3] complex rules (e.g. passenger way combinations). Furthermore, rules may have specific situations or time windows when they are applied or when explicitly not.

good solutions to cope with uncertainty, and to extend the knowledge-base (e.g. the rules). Unfortunately, in order to consider a large number of rules, similar to the mathematical programming approach, computing time becomes long.

Yan and Tang (2007) recognize that previous approaches<sup>38</sup> have not addressed the dynamic nature of gate assignments. So in their comprehensive work they developed a heuristic approach and tested it successfully at Taiwan Taoyuan International Airport (formerly, Chiang Kai-shek International Airport). The main components of their model include a stochastic flight delay gate assignment model for the planned gate assignments, a reassignment rule to be applied in real-time operations and two penalty adjustment methods (e.g. one for passenger waiting time). Their work is partly based on previous work of Yan, Shieh and Chen (2002) who addressed stochastic flight delays, and flexible buffer times<sup>39</sup> in the context of gate assignments.

Figure 12 summarizes and structures the main aspects of the above discussion.

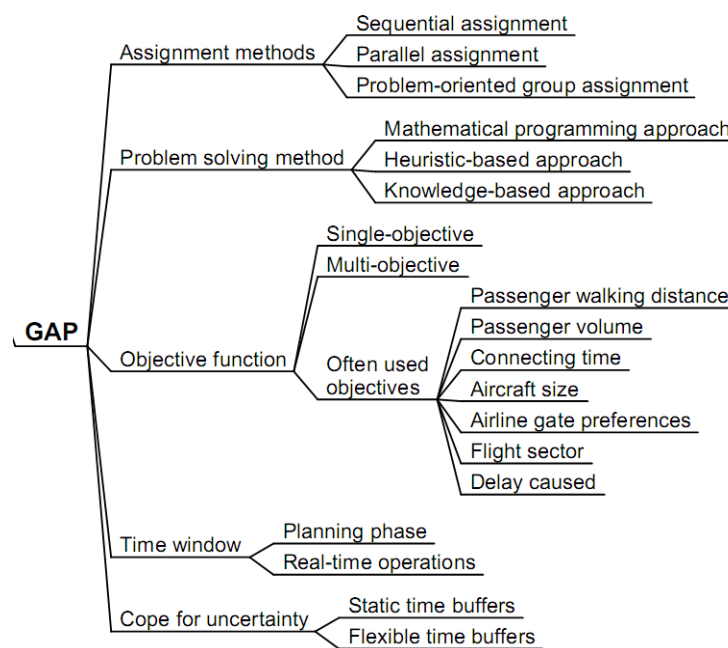


Figure 12: Structured body of knowledge of the gate assignment problem (GAP).

It is observed that previous approaches purely focus on operations, and that they do not show any application in commercial airport processes (e.g. retailing). Using

<sup>38</sup> They mention: Braaksma (1977), Babic et al. (1984), Mangoubi and Mathaisel (1985), Vanderstraetan and Bergeron (1988), Bihl (1990), Zhang et al. (1994), Cheng (1997), Yan and Chang (1998), Haghani and Chen (1998), Bolat (1999, 2000), and Yan and Huo (2001).

<sup>39</sup> Earlier works of Hassounah and Steuart (1993), Yan and Chang (1998), and Yan and Huo (2001) already used buffer times to address the delay aspect. However, those buffer times were fixed.

deterministic approaches, a large number of both gates and flights is not believed to be computable in an appropriate amount of time<sup>40</sup>. This paper also tries to contribute in that respect.

### 2.3.1 Use of a resource management system (RMS)

As mentioned above, gate allocation may be supported by means of information technology. Depending on the amount of traffic that needs to be handled at an airport and on the level of IT integration, it is usually a resource management system used for that task. Many of the systems on the market are able to account for different types of resources (e.g. check-in counters, gates and stands, baggage-carrousel). Some of them allow for defining dependencies between the types of resource. According to the author’s experience only such a holistic view copes for the high process integration at an airport.

However, the main objective in the use of an RMS<sup>41</sup> is to reduce the overall effort in the allocation (planning) process (, or in a situation of dense traffic to enable allocation at all). As indicated in Figure 13, the use of an RMS may help to reduce the effort of manual allocation planning.

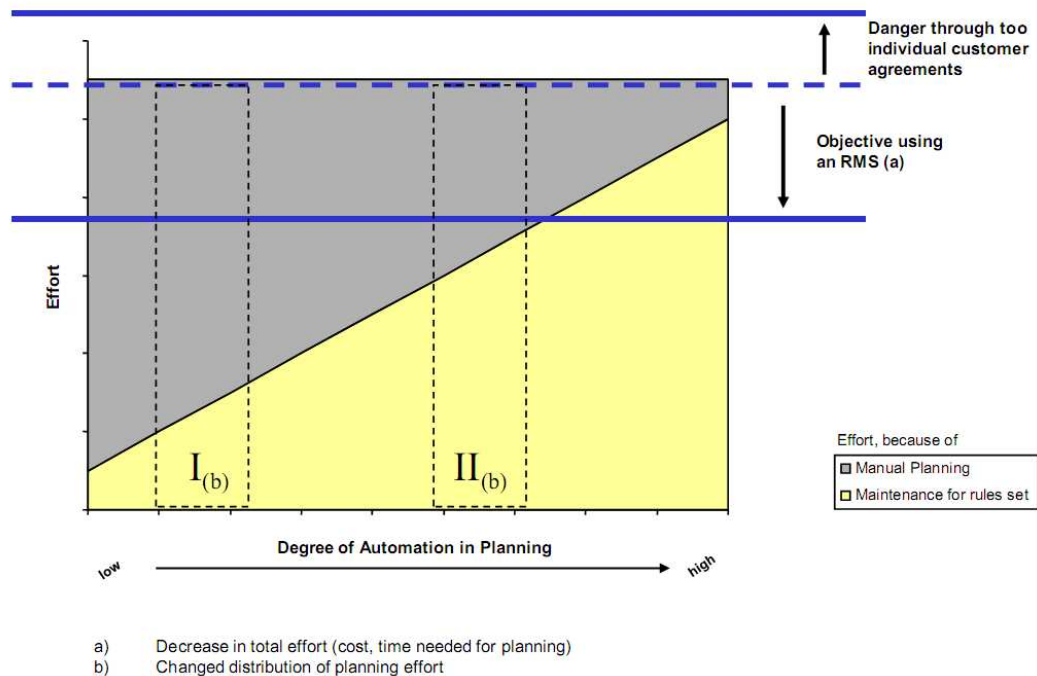


Figure 13: Resource management system (effort distribution 1).

<sup>40</sup> Neither for real-time operations, nor for planning phase.

<sup>41</sup> Here ‘RMS’ is to be understood as a gate allocation system.

Usually, such a system allows for definition of rules<sup>42</sup> in the allocation planning process. Depending on the degree of automation, the distribution of effort will change. The more automation it is enabled in the system, the less manual planning effort there will be necessary. This effort highly depends on the system’s capability to define or formulate allocation rules and on the complexity of such rules. It is upon the allocation planning staff to find the most suitable ‘mix’ between manual and automated planning. Other determinants in this question are the required update frequency and the time and resources given for the allocation planning task. For example, the effort distribution [I<sub>(b)</sub>] in Figure 13 describes a situation that is mainly handled by staff with little RMS support. Whereas [II<sub>(b)</sub>] heavily builds on systems support, and less manual effort is required. Ideally, a situation of reduced overall effort [(a)] is to be achieved. As mentioned above, this sort of ‘break-even-point’ in distribution effort has to be determined by the allocation planning staff. Usually, there is a danger of ‘over-automation’ trying to incorporate each and every exception into the system’s rule-set. The maintenance effort will increase and the rule-set may become logically inconsistent. This in return leads to increased manual effort at the time of operations where such inconsistencies may be discovered and (then under time pressure) coped for. In case of non-discovery, the service level provided would be reduced or in worst case a safety-critical situation may arise.

However, more often a situation is found as shown in Figure 14.

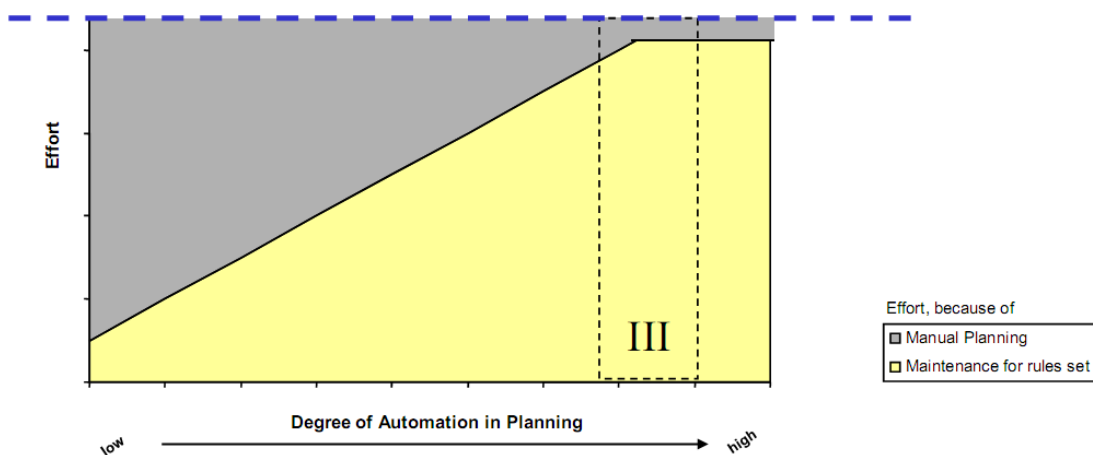


Figure 14: Resource management system (effort distribution 2).

<sup>42</sup> This forms part of the knowledge base in expert systems.

The distribution of effort as indicated in [III] describes a highly automated allocation planning process with economies of scale in the rule definition task. For example, this might be the case upon adding an additional carrier to the planning task, belonging to an airline alliance that has already been modelled in the rule-set. The same would apply to an additional frequency of an existing flight (e.g. from 4/7 to 5/7).

As described above, many aspects need to be taken into account using a resource management system. However, an RMS as such is no guarantee at all to improve the gate allocation process.

*[ALIGNMENT ASPECT]*<sup>43</sup>

*The alignment of the IT function and the business function also depends on proper IT support for business processes. The better an IT solution addresses the real conditions in the business, more suitable it is. An RMS that considers many business objectives and thus incorporates expert knowledge is well placed to support the business process in question. Therefore, a process-tailored piece of software seems to foster alignment (compare, Figure 10, (3) and (5)).*

Finally, there is the field of airport retailing that is discussed in the next chapter to provide sufficient background for the research project.

---

<sup>43</sup> In order to establish and maintain a strong link to the alignment aspect, explicit notes headlined ‘*[ALIGNMENT ASPECT]*’ are provided for the reader.

## 2.4 Airport retailing

Although existent since 1947<sup>44</sup>, from an academic perspective, airport retailing was widely neglected before the 1990s (Freathy and O'Connell, 1998c).<sup>45</sup> However, with the ongoing privatizations in the airport industry pressure on airport charges became a common scenario and thus new sources of revenue have been sought. So already in 1996 the commercial offer at airports contributed between 36% and 56% towards the overall income (ACI Datamonitor, in Cerovic 1998, p.12). In Europe in 2006 the spread was between approx. 22% and over 60% (study of annual reports, presented as chart in Graham, 2008). Usually, the more passengers an airport caters, the higher that share seems to be.

The commercial income (usually referred to as non-aeronautical revenue) is often spread into the categories of: retail, car parking, car rental, property, advertising, and other. Within the 'retail'-category the following sub-categories are found: duty free and tax free, currency exchange, food & beverage, specialty retail concessions (airside and landside), and other (amongst others: Doganis, 1992; Freathy and O'Connell, 1998c, Graham, 2008).

In addition to the abovementioned reason of new pressures in the course of airport privatizations, and along with crises that hit the air transport industry, there might be another reason why not much academic literature exists on airport retailing:

The general business function of 'retailing' has been widely explored by researchers from many perspectives. Hence, much of that knowledge in form of concepts, methodologies, paradigms and so forth has been simply applied to the airport retailing business as well. Airports seem to be 'just another place to sell goods and services'. Fernie (1995) regards airport retailing as a niche market, but with high spending customers. He compared the socio-economic classifications of passengers at U.K. airports with customers at the Metro Centre, Gateshead (near Newcastle Upon Tyne, U.K.) and discovered the high sales potential within typical passengers. So, in consequence the well-known concept of customer segmentation has been applied to the airport business by many researchers and practitioners as well (Freathy and O'Connell, 1998c; 2000a; Omar, 2001; Kim and Shin 2001; Geuens et al. 2004; Davitt 2005; Appold and Kasarda, 2006). Typical segments are (Freathy and O'Connell, 2000a):

---

<sup>44</sup> 1947 Shannon (Republic of Ireland) became the first airport with a duty-free shop (Freathy and O'Connell, 1998c, p.7).

<sup>45</sup> Nevertheless, much knowledge is present in form of conference presentations or market studies.



- domestic vs. international vs. transit
- short-haul vs. long-haul
- scheduled vs. non-scheduled
- business vs. pleasure
- intra-EU vs. non-intra-EU

Geuens et al. (2004) developed a passenger typology consisting of three classes (mood shoppers, apathetic or indifferent shoppers, and shopping lovers). As main shopping motivations for profiling they suggest factors that are: airport-related, atmospheric, experiential, and functional. Bork (2007) observes changed reasons to buy in an airport environment: from past reasons like cheap prices, rational reasons and a uniform product offer, he characterizes that modern airport shoppers:

- try to make bargains,
- seek the emotional buying experience,
- expect a variety of fancy products,
- lack time,
- are well-informed,
- and that their demographic background is today much broader, due to decreased fares.

Within their work on market segmentation in Europe Freathy and O'Connell (2000a) determine the following factors in their 'propensity-to-buy-function':

- tax environment (both direct and indirect taxation in the country of destination);
- lifestyle (culture, social class, disposable income, leisure time available);
- product types (merchandise mix, range and depth, number of branded goods available);
- retail environment (ambience of the airport, accessibility to retail outlets, store design and layout, staff attitudes and product knowledge);
- perceived value (the utility that accrues to the individual by purchasing or owning the product).

The more mature the airport retail business becomes, the more concepts will be tried to transfer from conventional (high street) retail market to the airport environment. For example, in many airports around the globe can be observed that Hildebrandt's (1988) '3-factor-model' of store image is successfully applied.

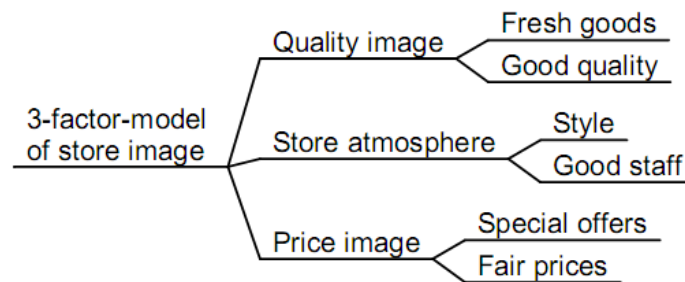


Figure 15: Basic structure of Hildebrandt's (1988) 3-factor-model of store image (illustrated by author).

Another known area from conventional retailing applied to airport retailing is that of impulse purchasing. Omar (2001) carried out a study asking 252 passengers and concludes that only when a passenger believes that acting on an impulse is appropriate, he or she shops. But what stimulates an impulse in the airport environment? Crawford and Melewar (2003) reviewed this aspect in more detail. They find the following airport impulse purchasing stimuli (p. 93):

- Value driven
- Holidays
- Gift giving
- Guilt
- Reward
- Occasion driven
- Forgotten items
- Confusion
- Exclusivity
- Disposal of foreign currency

In order to address this in airport retailing, they suggest an impulse-strategy-formulation, which contains the following elements:

- Reduce stress and anxiety
- Induce browsing
- Reduce normative traits
- Pure impulse

In a more comprehensive study of Entwistle (2007) more than 30,000 passengers from 20 airports constitute the basis for his postulation to conduct passenger segmentation for each part of an airport, and apply it to retailing and passenger flow control.

However, most contributing factors in most pieces of research are the number of passengers<sup>46</sup> along with their nationality (comprising many socio-cultural aspects).

<sup>46</sup> Appold and Kasarda (2006) in a detailed analysis of 75 U.S. airports.

Entwistle as well as other researchers (Freathy and O'Connell; Bork, 2007; Buendia and de Barros, 2008; Graham 2008) acknowledge that apart from the passenger profile, it is the dwell time throughout the passenger process that very much contributes towards the retail success.<sup>47</sup> This can firstly be explained by the time available to spent money and secondly by the influence on a passenger's mood (or stress level). In a very broad sense the dwell time on airside is determined by the processing at security checks (inflow) and by the boarding process (outflow). An adjacent area (not covered here, but in industry), which directly contributes to the dwell time issue is the management of the passenger (flow) process. Initiatives like 'simplifying passenger travel' (SPT), try to design the process in a way, that a passenger may pass through the airport terminal as seamless as possible. This is both: good for retail, because the shopping mood may improve, and it is bad for sales, because a seamless (and predictable) flow may lead to a later arrival at the airport. And in case the amount of time a passenger arrives later is more than the amount of time a passenger saves because of SPT-improved flow, the dwell time is likely to decrease.

Another area from classical retail that entered into discussion is the support of the (airport) retail supply chain (Freathy and O'Connell, 1998a). The main area of potential is to better satisfy demand with (just-in-time) supply. In the airport environment this is even more import (compared to high street), because storage space is scarce and expensive, and secondly quick reaction to anticipated (but changing) traffic flows may help to increase sales result considerably.

A core element within the IT support of the supply chain is the electronic point of sale (EPOS). This is (together with information from the passenger) a huge lever in market intelligence. More advanced retailers (and airports) use the data gathered from EPOS devices (cash-points, pay-terminals) to profile and analyse passenger purchases. Usually, information like the items purchased (price and quantity), and the flight number are stored. With introduction of 2-D bar-coded boarding passes the information gathering has become even easier. Theoretically, even more information may be generated from that.<sup>48</sup>

---

<sup>47</sup> Usually: higher 'footfall', 'penetration-rate' and 'conversion-rate' when passengers have sufficient time.

<sup>48</sup> As the airline possesses the passenger name record (PNR), any purchase may be drilled down to an individual. So, in case of many purchases per individual for a single flight event, a (personal) way tracking profile may easily be generated. As a next step the passenger information would need to be deleted and an anonymous set of passenger tracks through the terminal building would be available. For pure tracking purpose the PNR is not even necessary.

Two more areas of research in airport retailing were identified:

Firstly, reactions to the abolishment of duty free within the European Union were addressed by Freathy and O'Connell (2000b). And secondly, the management aspect itself (relationship: airport operator, retailer, supplier) was studied. Freathy and O'Connell (1998b) looked at the buying function in airport retailing with focus on:

- Buying structure in airport management retail operation
- Buying structure for retailer with domestic and airport stores
- Criteria for selection of new suppliers
- Supplier evaluation
- Effectiveness and efficiency of buying function.

Kim and Shin (2001) revealed in their survey that the commercial involvement of an airport in retail operations is one of the success factors for airport retailing.<sup>49</sup> The commercial involvement comprises aspects like:

- Wholly-owned subsidiary
- Direct operation
- Direct lease
- Joint venture
- Fee management contract
- Master concessionaire
- Developer approach

They find that the master concessionaire approach is “the most appropriate method of managing airport concessions.” (Kim and Shin, 2001, p.149)

---

<sup>49</sup> Other success factors mentioned are: total traffic handled, total amount of space allocation (in connection with layout and location), passenger characteristics, characteristics of contracts and rental fees, marketing strategy, and pricing strategy. For a view on marketing strategy in airport retailing see also Bork (2007).

The above is seen similar at Frankfurt Airport. Figure 16 provides a further view on potential drivers of airport retail success in respect to the influence that the airport operator in the role of a master concessionaire has on its concessionaires:

|               | External Driver                 | Internal Driver                    |                                 |
|---------------|---------------------------------|------------------------------------|---------------------------------|
| high impact   | legal restrictions              | passenger number                   | tenant mix                      |
|               | economic situation              | passenger mix                      | conditions of lease contract    |
| medium impact | currency fluctuation            | amount and quality of retail space | quality of shopping environment |
|               | consumer behavior               | passenger process/dwell time       | performance of tenants          |
|               | purchasing power                |                                    | sales promotion                 |
|               | taxation                        |                                    | marketing                       |
|               | price level in downtown markets |                                    |                                 |
|               | inflation                       |                                    |                                 |
|               | no control                      | low control                        | high control                    |

Figure 16: Drivers to retail success - Control and impact. Source: Fraport AG.

However, high impact drivers on retail success like passenger number and passenger mix, as well as the medium impact driver of passenger process/dwell time are to some extent under control of the airport operations function. Although the foremost control on passenger number and mix is executed by the airlines, the airport should try to benefit from that ‘general passenger offer’ and in consequence tailor it to maximise spending potential.

Similar to collaborative decision making initiatives (CDM), the question of a common, shared goal between the airport operator, the airlines and the retailers arises. Who shall benefit from retailing activities? Besides the service level aspect there is the question of single till vs. dual till. What is considered in airport revenues what will enter –or cross-subsidize– airport charges? Munich Airport for example established a close relationship with Deutsche Lufthansa in a way that for Terminal 2 they coordinate all aviation and non-aviation activities based on their respective core competencies. The financial result is shared. (Kerkloh, 2007).

The author of this paper observes that in most cases those areas of conventional (high street) retailing that needed adaption to the airport environment (because of an airport’s

characteristics or ongoing changes) have been subject to academic discussion. Other retail-related areas may be systematically explored for application at airports.

For example, technology is seen to further contribute in many fields:

- Automated supply chain management
- Information provisioning of the passenger (e.g. flight status)
- Information gathering about the passenger (e.g. upon purchase or within passenger flow)
- Information provisioning of the retail outlets (about schedules and passenger demographics)
- Information provisioning of airport operator by retailer (purchase data)

Or from a functional view: in operations, planning, and strategy (business intelligence).

#### *[ALIGNMENT ASPECT]*

*A precondition for alignment of the IT function and the business function is that the processing of information in any form supports the business processes. The airport retail sector seems to have many information processing aspects that may be supported by the use of suitable IT (compare, Figure 10, (2)). Another issue has to be placed in the alignment aspect: the output of the retail function is measurable in form of retail sales figures. So, the element of performance measurement can easily be addressed (compare, Figure 10, (6)).*

So far, the areas of business alignment of IT, of gate allocation (gate assignment problem) and that of airport retailing have been reviewed.

In each area useful ideas have been identified that can be further explored within this research project. Nevertheless, it is emphasized that a specialized view on a single topic may produce promising results, but usually fails upon transfer into practice. Therefore, an integral view is suggested.

## **2.5 Integral view and research question**

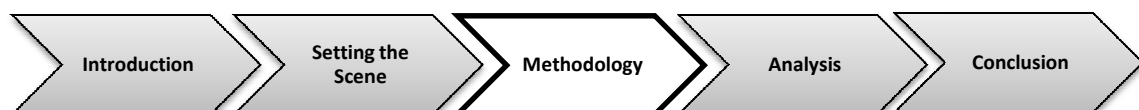
Supporting Pollalis' (2003) view, in this paper the role of IT in the airport business is understood as part of a well-integrated organizational system spanning over enterprise boundaries. The relationship between the business partners shall be governed by functional integration, information sharing and collaboration along the value chain(s).

Taking the above discussion into account, the research question is composed of (AIM) and (H.1), and formulated as follows:

*Does it seem feasible to increase retail sales at airports to a certain extent when applying specific criteria to the gate allocation process?*

In order to answer the research question, a suitable methodology has been worked out. The next chapter explains the approach and its elements in detail.

### *3. METHODOLOGY*





### 3. METHODOLOGY

To address the research question, it is essential to break down such a task into its elements and plan its stepwise accomplishment. The methodology aims to present a detailed view on the business processes in focus, develop the conceptual research model, identify the data needed, and to refine the model. To be able to quantify a possible output of the model, it was essential to develop a gate allocation algorithm.

A detailed view on the improvement approach and the self-developed simulation environment finalize this chapter.

#### 3.1 Choice of methodology (Justification)

The chosen approach of *systems thinking* with its associated methods allows to *decompose the system into its individual elements and to develop an idea about the ideal solution. This aims to design (synthesize) the elements and their relationships in such a way as to optimize the system as a whole regarding its achievements of objectives.*<sup>50</sup> Those elements found in a business context are to be understood as business processes producing a certain output. Much of that output depends on information as an input. At the same time the output can be information itself. Thus, strong emphasis has been put on information processing within the examined business processes.

The following chapters will choose business processes, which form part of an empirical analysis with much character of operations research. However, this is just used to produce an output on the business side, which might differ from the output observed without that form of IT support. So, the example explored can be seen as a vehicle to demonstrate that application of the various aspects of business/IT alignment may produce improved results on the business side.

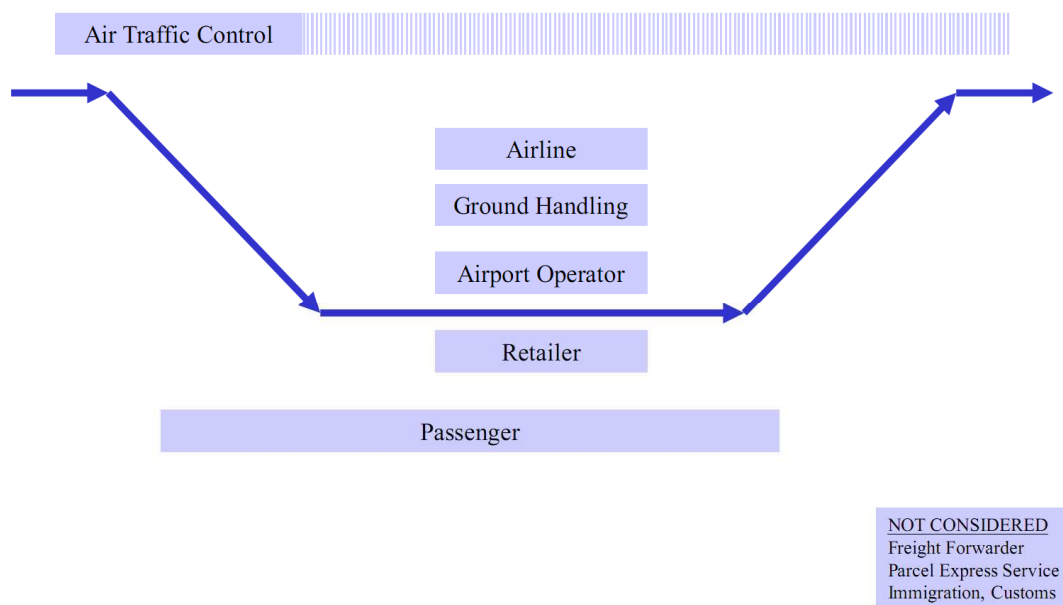
#### 3.2 Business context and scope

The airport business is a complex and highly regulated environment. To determine a source of potential benefits, all processes highly dependent on information processing are subject to further investigation.

---

<sup>50</sup> According to Koreimann (1995, p.7), in a translation from German by the author, in Klann (2001, 7). Also here the expression 'to optimize' is to be understood in the sense of 'to improve'.

Figure 17 shows the examined business context from a broader perspective. Depending on an airport operator's business model and its level of vertical integration, there might be a shift of functions from one business partner to another.<sup>51</sup> An example for this shows the field of ground handling operations. They can be conducted by the airport operator, one or more airlines, or else by one or more ground handling companies. Also a mixture of the aforementioned can be found.



**Figure 17: Business context of research project.**

Regardless of the organizational setup concerning the business partners involved, the flow of information and the need for information to perform the business functions remain constant. Changes in organizational boundaries (both within an organization and between different legal entities) will reflect in the necessity of interfaces between them, without superseding the need for a proper information flow. Consequently, the business context shown in Figure 17 is valid for various airport business models. However, in practice it requires great efforts between organizations agreeing on the use and exchange of information (also on a technological basis).<sup>52</sup>

The 'information intensity' model shall help to identify those processes in the business context carrying a higher improved output potential.

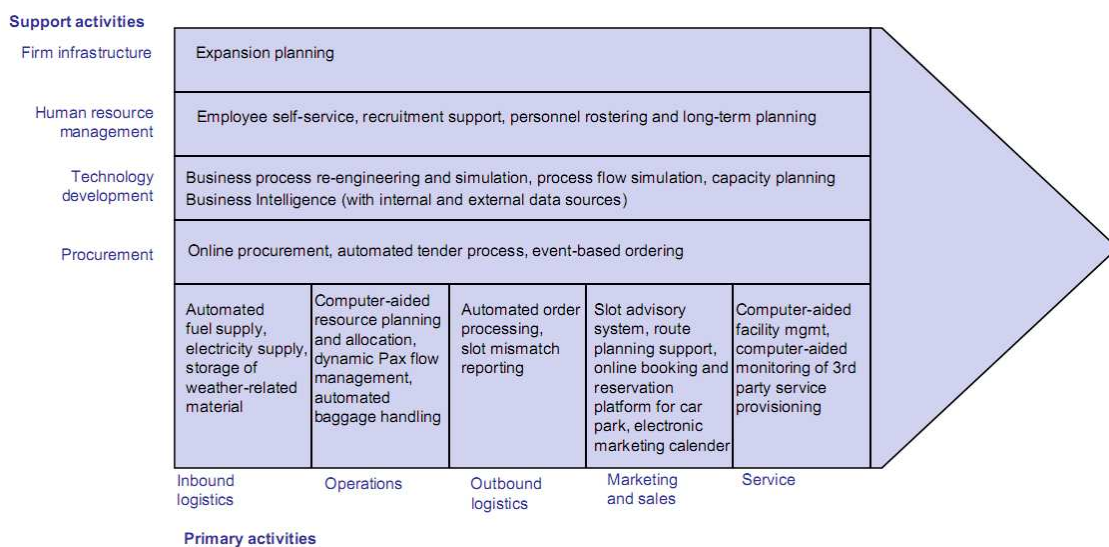
<sup>51</sup> So are passengers increasingly important as a customer for airport authorities (compare Jarach, 2001).

<sup>52</sup> The latter is usually spoken of as Electronic Data Interchange (EDI), but it has to be mentioned that also *within* a legal entity such information transfer is regulated and technically implemented. In that case it is referred to as data integration, often as part of Enterprise Application Integration (EAI).

**3.2.1 ‘Information Intensity’ model applied to airport business**

According to the model of Porter and Millar (1985), competitive advantage lies in the intensity of its processed information, both within a firm’s value chain and within a firm’s products. Consequently, in a first step an airport value chain has been modelled following the original concept of Porter (1980).

Figure 18 shows that a variety of activities within the value chain depend on information processing.<sup>53</sup>



**Figure 18: Value chain of an airport with focus on use of technology.**

To gain competitive advantage based on the value chain, Porter and Millar (1985) suggest the following steps:

1. Assess information intensity
2. Determine the role of information technology in industry structure
3. Identify and rank the ways in which information technology might create competitive advantage
4. Investigate how information technology might spawn new businesses
5. Develop a plan for taking advantage of information technology

Of particular interest to the research topic is the assessment of information intensity and the last point in a form of suggesting a possible application of the findings.

As a vehicle to assess the information intensity, a corresponding matrix has been applied to the airport business (Figure 19).

<sup>53</sup> Compare also Albers et al. (2005), Fig. 2, which has a slightly different focus on the value chain.

Matrix determination indicators according to Porter and Millar (1985):

For potentially high information intensity in the value chain:

- a large number of suppliers or customers with whom the company deals directly,
- a product requiring a large quantity of information in selling,
- a product line with many distinct product varieties,
- a product composed of many parts,
- [a product with] a large number of steps in a company’s manufacturing process,
- [a product with] a long cycle time from the initial order to the delivered product.

For potentially high information intensity in the product:

- a product that mainly provides information,
- a product whose operation involves substantial information processing,
- a product whose use requires the buyer to process a lot of information,
- a product requiring especially high costs for buyer training,
- a product that has many alternative uses or
- [a product that] is sold to a buyer with high information intensity in his or her own business.

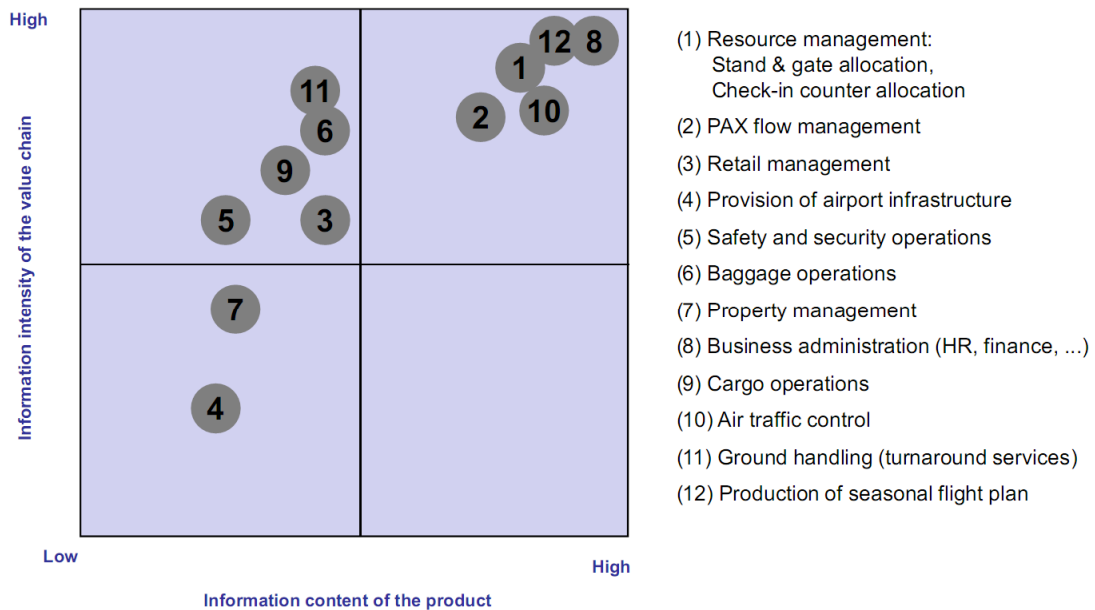


Figure 19: Information intensity matrix, applied to airports.

Once the information-intensive processes have been identified, they can be analysed in detail.

The basis of this paper had to be a highly information-intensive airport process with a potential influence on a commercial airport process.

The processes (1), (2), (8), (10) and (12) shown in Figure 19 were potential candidates. Process (8) was taken off the list for not being an airport's operational process. Air traffic control (10) forms part of airport operations, however, it is usually handled by national authorities rather than an airport operator. Both processes (1) and (12) highly influence the commercial process of airport retailing as they are significantly involved in feeding retail with potential customers. The core difference between them is the time factor. The seasonal flight plan (process 12) is usually produced one to two seasons in advance. The stand and gate allocation process (1) spans within the current flight plan up to the actual hour of operation. However, the basic task of allocating flights (aircraft) to gates is the same in both processes. Therefore, a generic process of stand and gate allocation serves as a basis for further research.

#### *[ALIGNMENT ASPECT]*

*A core aspect in the alignment of the IT function and the business function is that business processes are supported, which are likely take advantage of information processing more than others (compare, Figure 10, (2)). Note: Other processes from Figure 19 could also have been taken for analysis regarding IT support and potentially different output. However, according to the information intensity matrix preference was given to the gate allocation process. In case of different processes chosen, major parts within the methodology chapter would need to address process analysis and output generation specific to those processes.*

### **3.2.2 Applicable airport environment**

As mentioned earlier, the airport type or its business model plays a certain role in the research context. This might be to an extent that it may not be the airport operator alone who would be in a position to control the relevant processes and the flow of information within the context looked at. Furthermore, it is important that there exists a retail offer at the airport in question and the allocation of flights to gates somehow determine which part of that offer can be used by a passenger. Thus, it is a basic prerequisite to apply the research project to an airport of decentralized retail areas with a diversified range of products. In case there would be a homogeneous or a centralized retail offer, from a sales perspective, there would be no difference in the location of gates that flights are

allocated to. So, one precondition for the research project's meaningful application has been identified.

A further prerequisite are constrained gate resources that serve a certain retail area; improved allocation is only necessary where there is insufficient terminal space within a favourable retail area.

The type of airport may have an influence on the overall result. Typical characteristics for airport classification are:

- Type: hub-and-spoke, origin & destination (O&D)
- Size: Small, medium, large
- Geo-political characteristics: national, international, continental, inter-continental

As retail sales depend on the nationality and number of passengers, highly frequented international (inter-continental) hub-airports provide the best setting for a large lever in increased sales. Nevertheless, highly frequented international O&D airports are no less likely to increase retail sales. Usually, duty-free shops are the drivers of retail sales predominantly generated upon departure.

For the sake of deeper insight, this paper breaks down the two business processes in question in a way usually applied within business process re-engineering (BPR). The notation used is called IDEF0<sup>54</sup>, one of the standard notations in BPR. Figure 20 outlines that the central element, an activity, is described by its input and the output it produces. There may be controls (e.g. rules, environment) restricting the activity to a certain extent. Mechanisms (e.g. tools) may be used as an aid for the activity to produce its output.

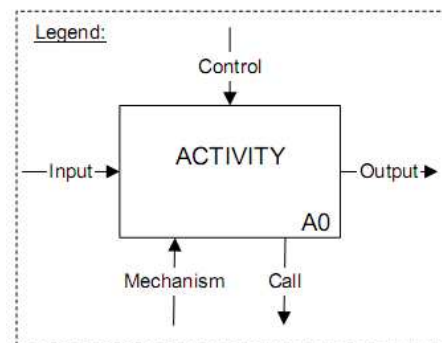


Figure 20: IDEF0 legend of an activity

Finally, an activity may start (call) concurrent activities or may invoke another activity, which in return produces output as an input for the calling activity (iterations or loops of processes may be modelled that way).

<sup>54</sup> IDEF0: **I**ntegrated **D**efinition (Originally stood for ICAM Definition. ICAM was the Integrated Computer-Aided Manufacturing initiative of the U.S. Air Force). It is a method to model functions, activities or processes in a structured way. IDEF0 is a member of a notation family used for processes, information, and simulation.

### 3.2.3 Business process decomposition for airport retailing

Using the methodology and notation introduced in the previous chapter, the process of airport retailing can be described as shown in Figure 21.

The main input to generate sales (and finally revenue) within the retail process is the number of passengers (PAX). The mechanisms that support that activity are of course the retail offering itself (retail, food & beverage, duty free), as well as means of information technology (e.g. systems for planning, selling)

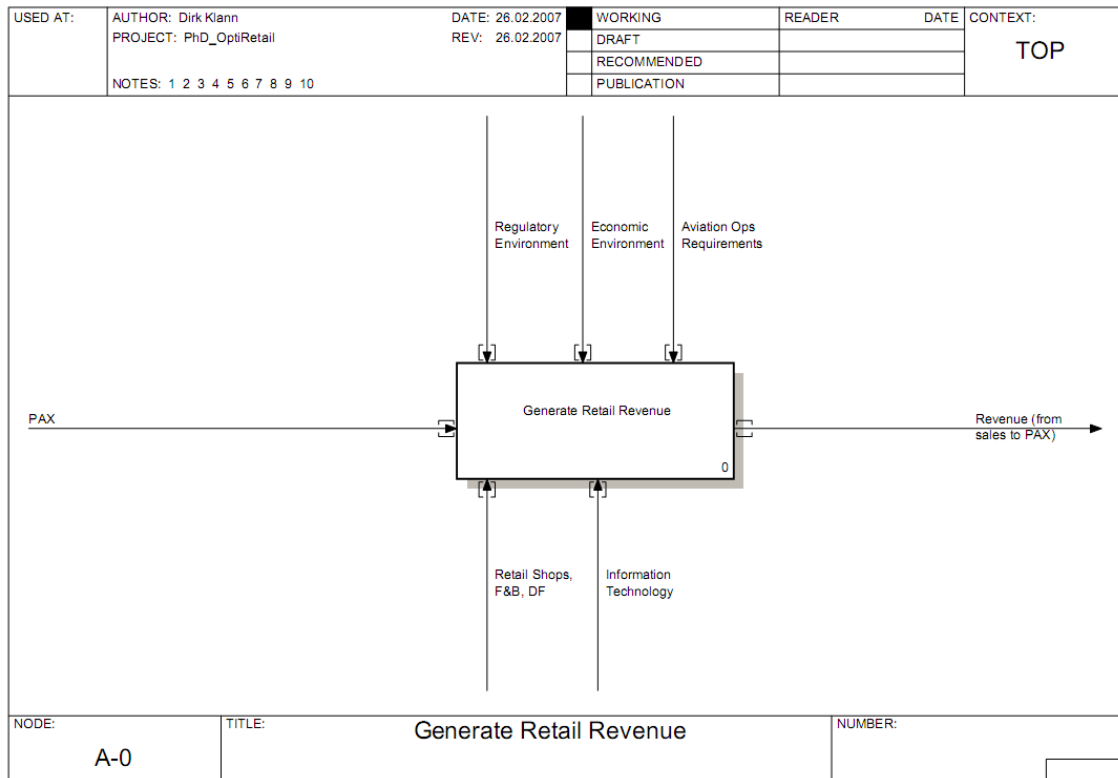


Figure 21: Decomposition of retail process, top level.

Unfortunately, there are many constraints that put a burden on the retail generation process. The retail business as such is already subject to many laws and regulations (e.g. competition law, law to protect minors). In addition to that, national trade law (customs regulations) applies, as well. Furthermore, the general economic situation (tax rates, currency rates, level of unemployment, etc.) constrain the process performance even more. This applies to both the country of the airport in question, as well as to a passenger's country of origin. Finally, a third source of constraints limits the processes' ability to perform favourable output: Requirements from an operational perspective. Those may be as simple as the application of safety procedures (e.g. wing tip clearance

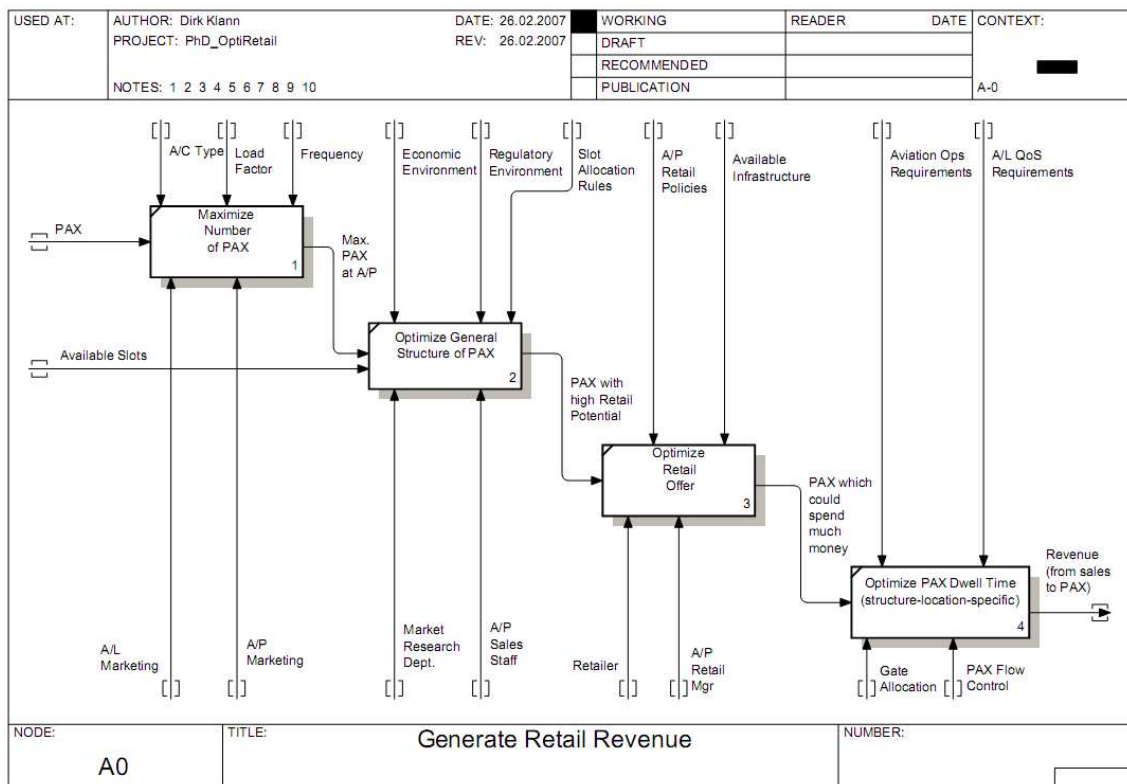
of aircraft, operational time buffers to cope for unexpected events, air traffic control permissions, or mandatory sequence of events in the turnaround process of an aircraft).

The task of retail sales generation can be compared with a standard task in logistics, and may be formulated as:

*‘Provide the maximum number of passengers of the right mix to a tailored retail offering in a relaxing atmosphere with enough time to spend money.’*

Consequently, at the next level of detail in Figure 22 the process has been broken down (decomposition) into four sub-processes (or activities):<sup>55</sup>

1. Maximize number of PAX
2. Optimize General Structure of PAX
3. Optimize Retail Offer
4. Optimize PAX Dwell Time



**Figure 22: Decomposition of retail process, 2<sup>nd</sup> level.**

<sup>55</sup> For clarification: the decomposition aims to provide a better understanding of the process. It is not aimed to improve the four sub-processes as such.



---

When analysing the retail process, it has to be mentioned that it needs to be looked at from two different time frames. Firstly, there is a planning phase in which you create the overall determinants for future business potential. For example, the destinations served by airlines determine passengers' origin to a large extent. Therefore, already in the phase of seasonal flight planning the basic structure of passenger mix is determined. The dimension of time during a day of operation is basically determined by the slots granted to airlines. Even earlier than a flight plan season ahead, the general retail offer (shop mix) determines to a certain level the potential in sales from retail. However, there is a second phase *within* the current flight plan season. Within that phase a couple of times, the seasonal plan will be refined (due to changes in schedules, for technical reasons or also for weather reasons). These iterations (or updates) continue up to the moment of operations. A major portion of the above mentioned task is about bringing passengers to the retail offer.<sup>56</sup>

However, having identified major contributing factors towards retailing, it is not aimed to maximize the number of passengers, to improve the general structure of passengers, to change the retail offering or to improve the passenger dwell time on an overall basis. In fact, the approach in this paper aims to improve the fit of passengers towards the existing retail offering in terms of their nationality (derived from a flight's country of destination).

As identified earlier, the second process investigated in this paper – gate allocation – is very much in a position to conduct that distribution task. Applying the same notation, the next chapter discusses it in more detail.

---

<sup>56</sup> Although it needs to be mentioned that also the retail offer itself may actively locate near favourable passengers (e.g. approaching waiting passengers in a queue with movable booths).

### 3.2.4 Business process decomposition for gate allocation

The same that has been mentioned above regarding timing (planning phase and then continuous updates until moment of operations) applies to the gate allocation process.

Figure 23 outlines the gate allocation process from a top level.

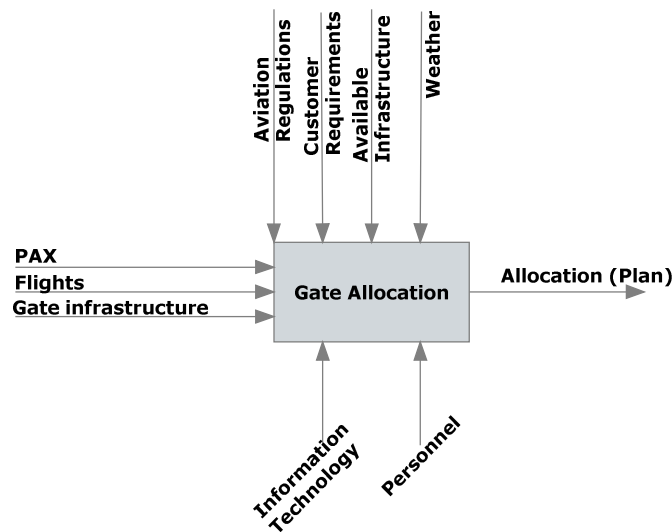


Figure 23: Decomposition of gate allocation process, top level.

The major inputs to the gate allocation process are the passengers, the flights (with their scheduled times) and the gate infrastructure. The latter might be regarded as a mechanism as well (like information technology and personnel), but because of infrastructure being an indispensable resource for the whole process, it has been taken as one of the input factors. The output of the gate allocation process is an allocation plan that satisfies the various requirements. Here the major stakeholder is the operations function. It is operations that actually conducts the allocation and tries to incorporate the majority of requirements<sup>57</sup>. Part of the overall infrastructure may not be usable for a specific flight, because of restrictions in terms of space limitations or simply due to maintenance in progress. Therefore, the remaining available infrastructure may be considerably less than the overall infrastructure. The improvement element within the allocation (planning) process usually tries to address this challenge.

<sup>57</sup> A typical airline customer requirement is usually to be allocated close to alliance partners, to a connecting flight, to an own business lounge or close to check-in facilities for a specific flight.

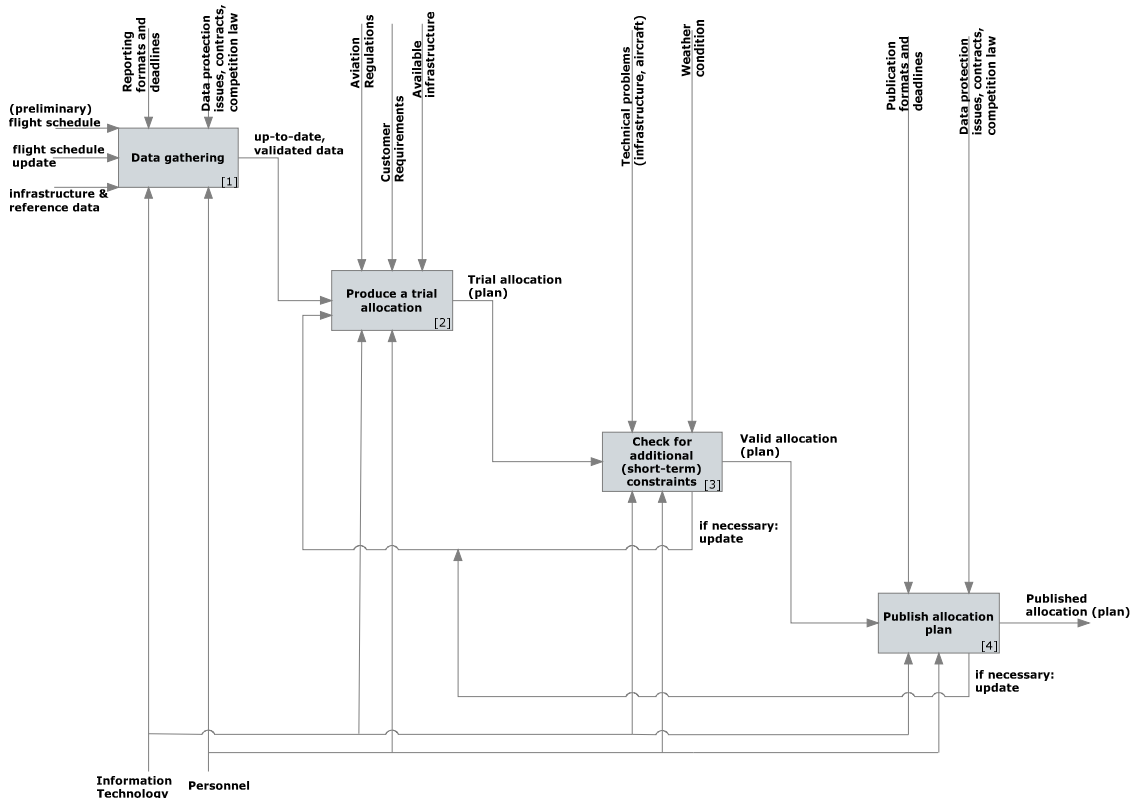


Figure 24: Decomposition of gate allocation process, 2<sup>nd</sup> level.

The closer to the day of operations the more updates will be due to technical reasons, due to weather, or due to other operational reasons. Therefore, the activities [1] through [3] in Figure 24 are usually run several times until the day of operations has been reached. As many business partners are involved in the whole process, especially in data provisioning, the quality of an allocation highly depends on the timely delivery of information. Regarding the output portion of the process (the allocation plan) many stakeholders depend on it for their own process (resource) planning. Those stakeholders include: immigrations, customs, ground handling services, security staff, outbound border control, airline check-in staff and retail outlets.

To this point, the major inputs and constraining factors<sup>58</sup> of the gate allocation process and of the retailing process are determined.

<sup>58</sup> ‘Control arrows’ in any IDEF0 diagram represent constraints. For the gate allocation process these are basically: Aviation regulations, customer requirements, available infrastructure and weather.

The business context and scope were described from a process point of view. However, as information is found to play an important role in value creation, the next chapter introduces an information model tailored to the business context. Again, the notation chosen is from within the IDEF family<sup>59</sup>.

### 3.2.5 Airport information model in context of business processes examined

Before the research context-specific information model is discussed, a typical purely operational view – similar to the IATA/ACI scenario (see Figure 9) – is presented by Kelemen (2005). The model supports the integration aspect and suggests a data flow as outlined in Figure 25.

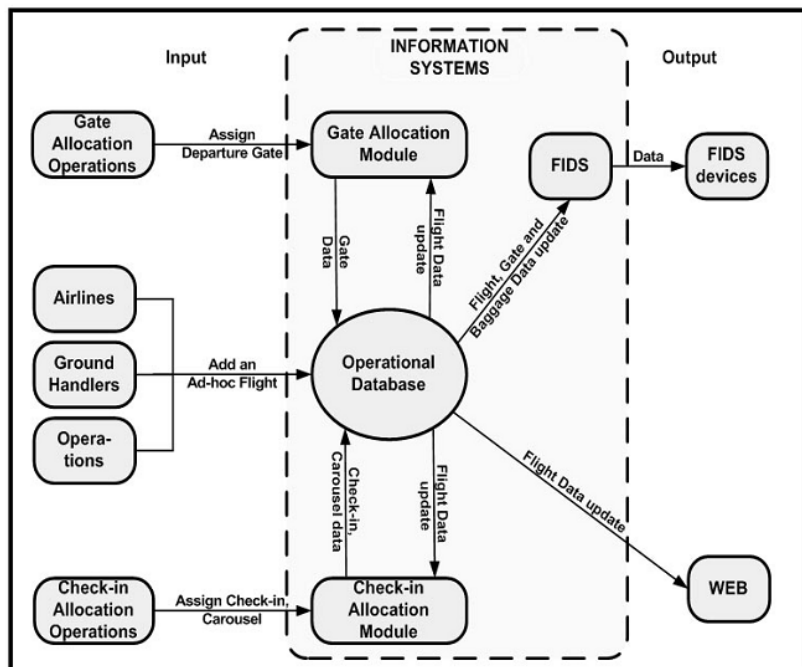


Figure 25: Resource Management Data Flow Chart (Source: Kelemen, 2005, p. 22).

This overview helps to identify the broader context and links that need to be in place for planning and operations, but for the research topic there are both: missing elements and unnecessary elements. Hence, this research project suggests an information model that focuses on only those entities that will play a role in later process of solution finding.

The business context described above comprises various players and two major business processes. However, as a smooth-running gate allocation process is a prerequisite for an improved retailing process, the information model emphasizes operational aspects within the business context.

<sup>59</sup> IDEF1X is member of the IDEF family and used for data modelling.



information model, consequently there cannot be an entry in FLIGHT EVENT without a corresponding entry within the FLIGHT entity.

So, the first is dependent on the latter. These types of relationships are expressed in the information model, developed as part of this research project. As outlined in Figure 27, a flight is mainly defined by its airline, the country of origin or destination, a possible alliance membership, and its ‘relative retail factor’.

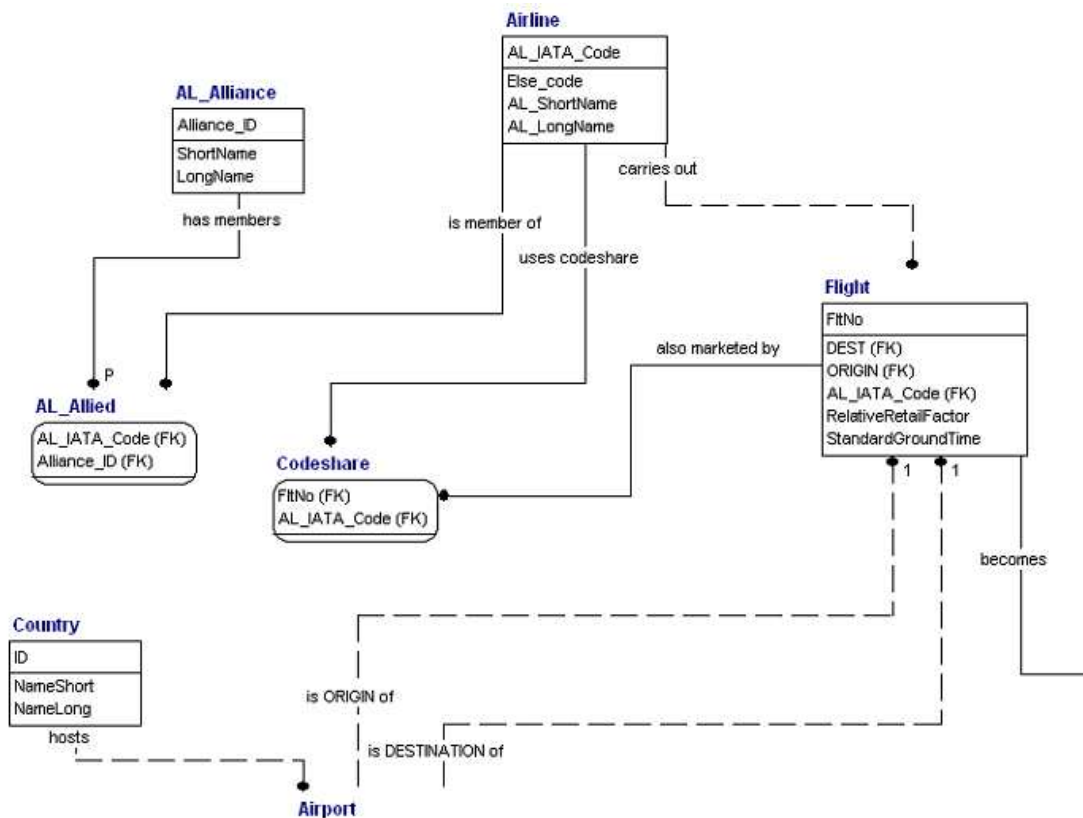


Figure 27: Airport information model, domain: flight.

The relative retail factor describes the spending behaviour of passengers on that flight in relation to other flights departing from the airport being investigated. It is also referred to as ‘relative retail-worthiness’.

The second major domain within the information model comprises all entities around the airport infrastructure.<sup>62</sup>

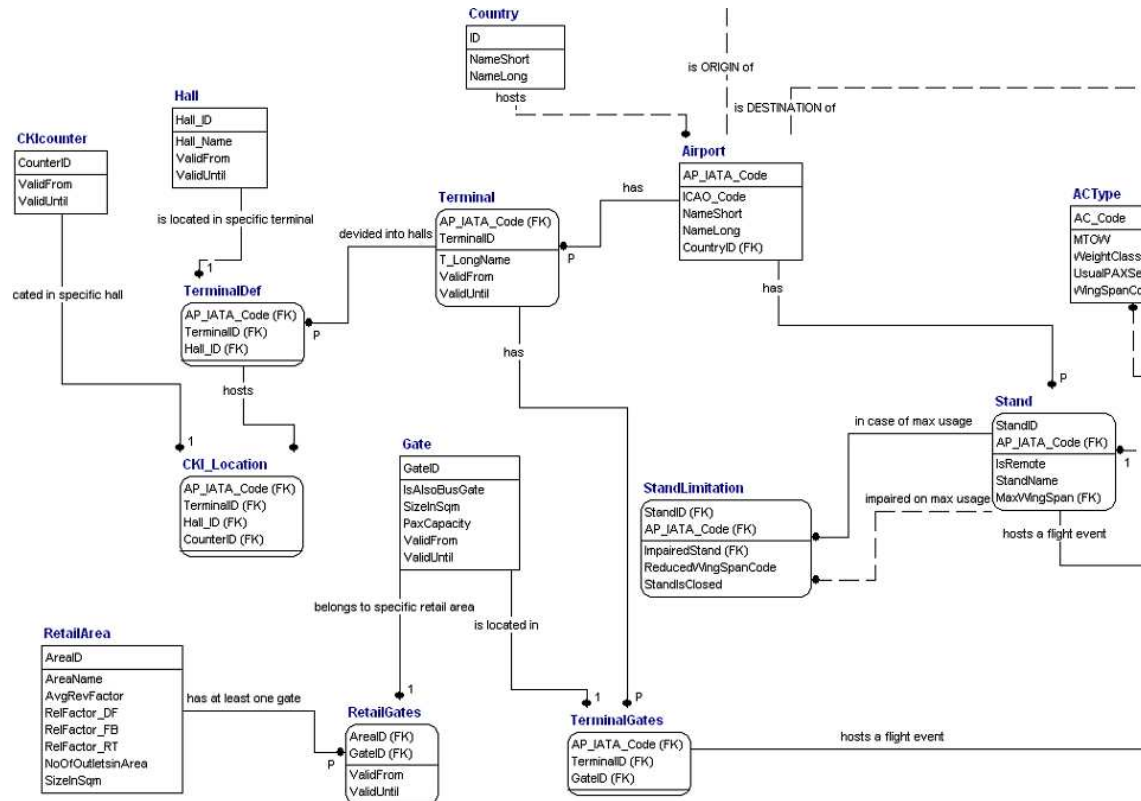


Figure 28: Airport information model, domain: airport infrastructure.

The entities modelled in Figure 28 describe that an airport consists of terminals, and belongs to a country. The latter is important because it determines the spending behaviour of passengers to a destination airport. This information is used in case no flight-specific spending information would be available. Terminal information is important to cope for alliance-specific gate allocation. In order to account for physical constraints both for aircraft and passengers, corresponding attributes have been incorporated in the entities of gates and stands.

To address overall retail performance, retail areas are described in terms of relative factors (in comparison to other retail areas at the airport in question). A retail area consists of at least one gate from where passengers may leave the terminal building to board a plane. The boarding process can be either directly into the aircraft (e.g. via a

<sup>62</sup> It has to be emphasized that only those entities (and associated attributes) have been incorporated into the model as found relevant for the research project. A comprehensive information model of the airport domain (incl. airlines and air traffic control) easily spans over several hundred entities.

bridge or via a stair case and short walk across the apron area), or more indirectly into a bus, which then transfers the passengers to the aircraft.

Finally, the information model describes the flight event itself. A flight event is based on all the reference information found in the aforementioned domains.

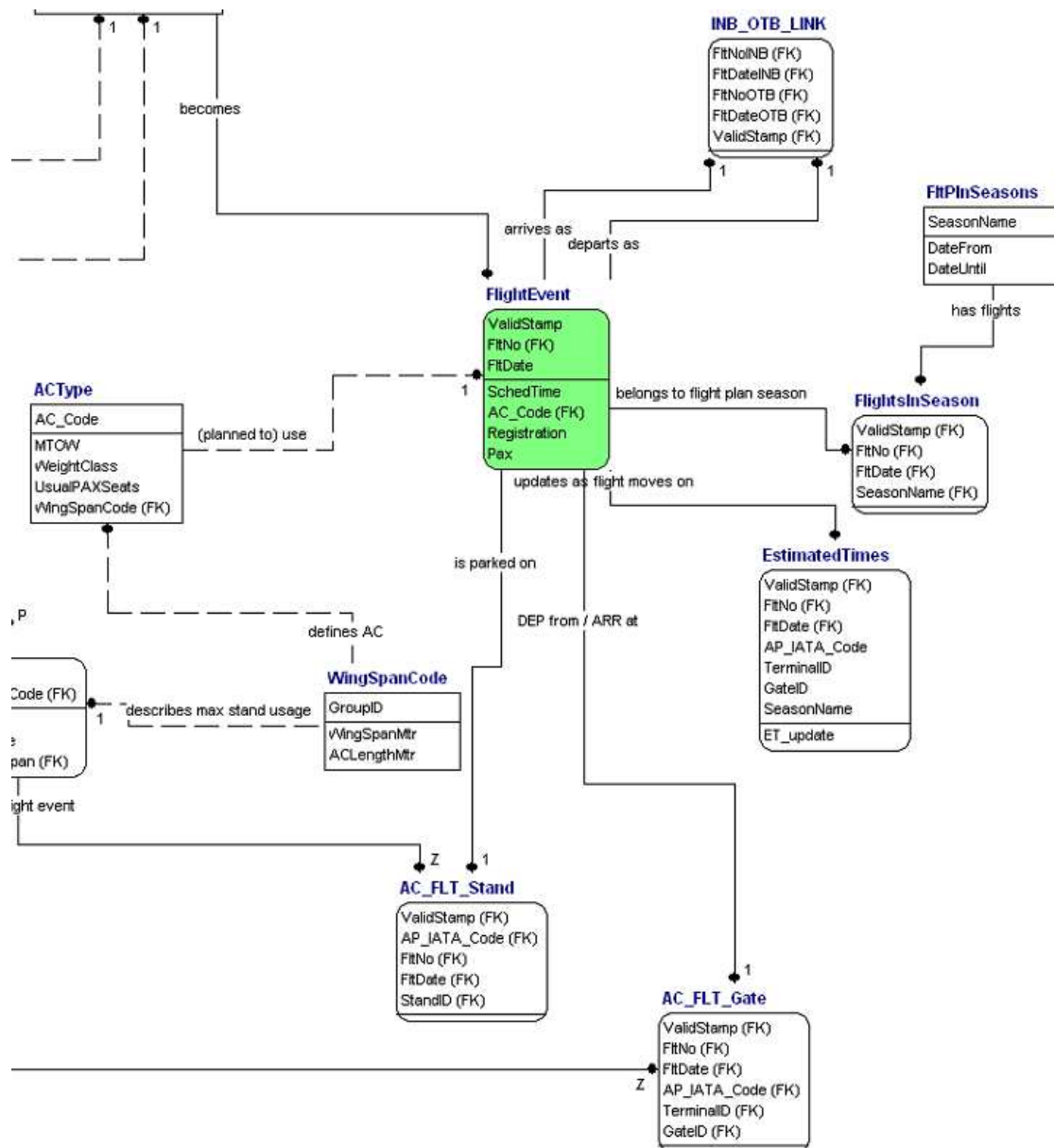


Figure 29: Airport information model, domain: flight event.

As to be observed in Figure 29 the most important characteristic of a flight event is that it contains current information about the conduct of operations. So, relevant information like time estimates, aircraft changes, but most notably stand and gate changes will be updated in this domain.



Up to this point, the business context has been described in general, as well as in terms of process decompositions and in form of an information model.

*[ALIGNMENT ASPECT]*

*The better (and usually more detailed) a business process and its use of information is known, the more likely it is that the IT function can produce a solution, which fulfils business requirements (compare, Figure 10, (3) and (5)).*

In the next chapter this foundation is used to develop a conceptual research model.

### 3.3 Formulation of conceptual research model

The conceptual research model aims to visually represent potential relationships between its elements. Further, it aims to outline where changes within the model might reflect changes in process output.

Basically, it describes two different ways, how the processes in focus may interoperate with each other. In the first set-up the gate allocation process is focused purely on operational goals. The process draws its information from its own data sources and produces output (performs) according to a rule set, which is defined for sole use of operations. Similar to that, the retailing process works on an own information basis and applies its own rule set to conduct business.

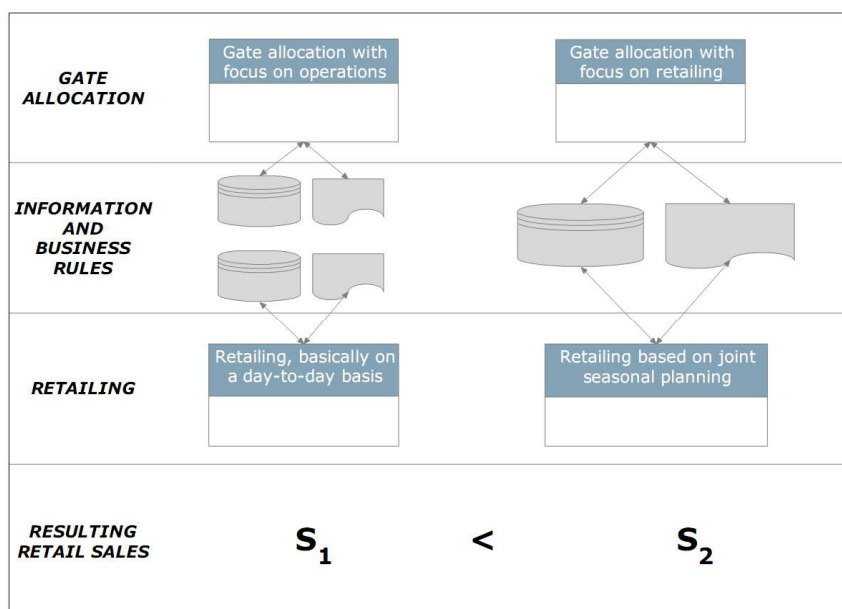


Figure 30: Conceptual research model with underlying statement (H.1).

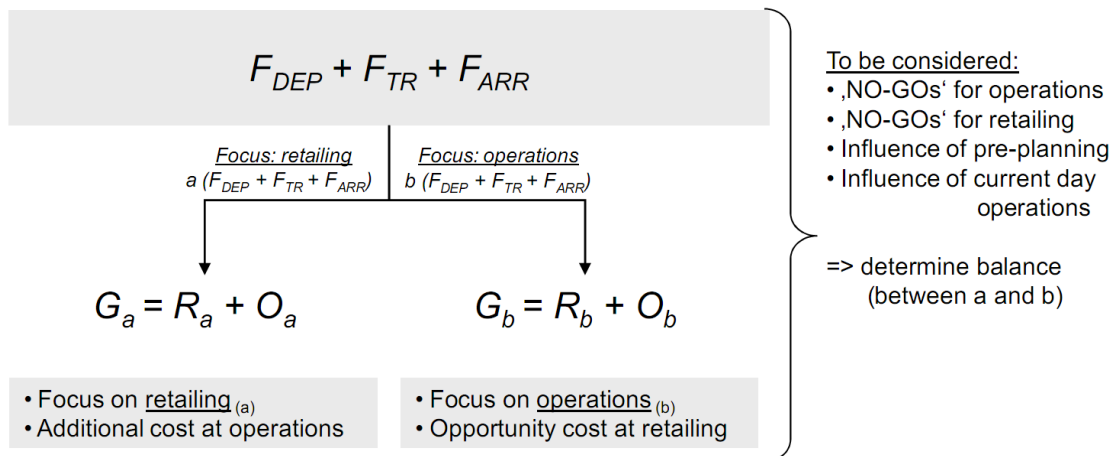
As no joint seasonal planning takes place here (situation leading to  $S_1$ ), retailing basically plans on a day-to-day basis. This means that no long-term customer-oriented offering can be accounted for in the supply chain. With a clear seasonal outlook of the flights to be expected in a certain retail area, this aspect may be addressed.

The major difference in the second set-up is that both processes make use of a shared pool of information, and most importantly perform according to a harmonized rule set. The latter copes for rules that are important for both functions, e.g. maximum number of passengers in a retail area at any time, or preference of specific flights in a certain retail area, based on their retail-worthiness. For each of these requirements meaningful values need to be agreed on. The sum of these agreements form the harmonized rule set.

In order to achieve such an aligned base of both information and business rules (as shown in Figure 30, in the **S<sub>2</sub>** column), there needs to be close coordination (amongst retailing and gate allocation) during the planning phase of a seasonal flight plan. Such a joint common approach allows for better harmonization of the retail offering and the supply with passengers.

Thus, in Figure 30 the basic underlying statement, introduced as (H.1) in Chapter 1.4, claims that retail sales could be increased, if gate allocation would focus on retailing. As mentioned above, this can be accomplished by a joint seasonal flight planning process based on shared information and a harmonized rule-set. The latter would need to incorporate only a minimum of hard constraints as to achieve high levels of freedom for the (retail-favoured) improvement purpose.

Nevertheless, it must not be omitted that the increase in retail sales may come at a cost on the operational side. The following puts the conceptual model into context.



Legend:

- F = Flight (DEParture, TRansit and TRansfer, ARRival)*
- a = focus on retailing (i.e. to improve retail result, by application of respective business rules)*
- b = focus on operations (i.e. to apply operational allocation rules only)*
- G<sub>a</sub> = Income applying a specific gate allocation (focussed on retailing)*
- G<sub>b</sub> = Income applying a specific gate allocation (focussed on operations)*
- R<sub>a</sub> = Income from retailing (focussed on retailing)*
- R<sub>b</sub> = Income from retailing (focussed on operations)*
- O<sub>a</sub> = Income from operations (fees/charges less penalties) (focussed on retailing)*
- O<sub>b</sub> = Income from operations (fees/charges less penalties) (focussed on operations)*

**Figure 31: Context of conceptual research model.**

As shown in Figure 31, the overall income generated from operations and retailing is based on total traffic. Within the research context only passenger flights will be looked at. At constrained airports this might be a factor to consider, because changes in gate allocation may result in higher traffic load on the apron and taxiways. This in consequence may have an impact on cargo flights, taxiing to and from the runway.

There is another simplification in the model. Retail sales will only be calculated for departing traffic (incl. the departing portion of transfers, and transits). There are two reasons for that. Firstly, there is only a very limited amount of sales produced by arrival passengers (almost no arrival duty free in Europe). Secondly, the retail sales data is purely based on departing passengers. Thus only passenger traffic  $F_{DEP} + F_{TR}$  enters the research model. The gate allocation algorithm will need to apply certain rules for the allocation process. These rules suggest some hard constraints ('NO-GOs') and soft constraints ('if-possibles') to be applied in the allocation planning process.

In a business environment the following questions would need to be answered:

- Does income from retailing set off additional cost at operations when allocated 'retail-friendly'?
- Are there ways to reduce opportunity cost at retailing when allocated 'operations-friendly'?
- To which extent is operations in a position to allocate flights in a retail-favoured way?

However, as a detailed view on additional cost at operations would be beyond the scope of this project, the conceptual model examines  $R_a$  (in Figure 31) in different scenarios and compares it to  $R_b$ . The latter is derived from the retail result that actual allocations would have produced. Those ('real-world') allocations had been conducted to achieve operational goals. This way a potential increase in retail sales ( $S_2 > S_1$ , Figure 30) should be determined or disproved. The main factors contributing to additional cost at operations will be discussed with the limitations of this piece of research in Chapter 5.4.

Having identified the processes' activities and the information structure of the research context, in a next step the specific information to be gathered will be described in more detail.

#### *[ALIGNMENT ASPECT]*

*The first alignment aspect here is that IT supports to integrate business functions (compare, Figure 10, (3) and (5)). Secondly, the aspect of performance measurement (retail sales) is incorporated in the discussion (compare, Figure 10, (6)).*

### 3.4 Identification of information / data needed

From the process decompositions and the data model, introduced above, the required information entities can be identified. In general, it has to be differentiated between infrastructural data<sup>63</sup>, flight schedules, retail sales data, and the rule set<sup>64</sup>.

In order to construct a test bed for the to-be-developed algorithm and simulation software, that ought to be able to cope with high loads of traffic, data from Frankfurt Airport was used. Here, two very important aspects needed to be considered:

Firstly and most important, a precondition regarding the use of data at all, was to make it anonymous to such an extent that it cannot be deduced towards individual actual flight numbers or airlines without explicit permission of the respective data owner. In consequence, all output generated by the software will be in terms of references (running index numbers), or aggregated to an extent that this information could be found in public as well. A third form of information used is that of relative index numbers (e.g. sales of a flight in relation to the overall average of sales, expressed as a factor or percentage).

However, in order to cross-check and validate the output, it would be possible to apply a sort of translation table to the output, so as to derive original flights numbers.<sup>65</sup>

Secondly, the data had to be prepared for usage in the research project. At this, a minor aspect has been to extract the data out of existing systems into file formats like flat text files or files in some sort of spreadsheet format. Far more work intensive has been another task within so called *data cleansing*, i.e. to achieve logical consistency and integrity.

A few examples may highlight that:

- (1) Codes for countries, airports and other reference data had to be standardized: Is the code for SPAIN = E, ES, SP, or else?
- (2) To cope for the factor of time: Some of the data sources had different definitions of the member states of the European Union, of former Yugoslavia, the former Russian Federation, and so on.
- (3) There have been departing flights with no passengers on board but massive sales.

---

<sup>63</sup> Also referred to as reference data.

<sup>64</sup> The rule set will be introduced in Chapter 3.4.4 and described in more detail in Chapter 4.2.2.

<sup>65</sup> With permission of data owners only.

- (4) To cope for cancelled flights or flights with an aircraft change after boarding.
- (5) In actual flight data there were flights, which did not exist in the corresponding seasonal flight plan.

Many more of those aspects had to be taken into account before using the data to describe the current (as-is) situation and to calibrate the research model.

Many of the above aspects could be addressed with simple SQL<sup>66</sup> statements. In that case flat file data was imported into a plain and empty MySQL database and worked on with AnySQL Maestro<sup>67</sup>. More complex data cleansing tasks have either been accomplished with self-written software routines (part of source code in Appendix A, Chapter 8.2.1) or in case of only few occurrences of change with an advanced multi-purpose text editor<sup>68</sup>.

The data cleansing task is both very important for sound results of an analysis and very time consuming.<sup>69</sup>

The following chapters describe the different types of data used for further research.

### 3.4.1 Infrastructural (reference) data

In order to describe the physical set-up of an airport along with the main players (airlines, retailers, passengers) in the research context, information had to be gathered about:

- the airport layout (terminal buildings, piers, gates, gate hold rooms, retail areas),
- aircraft data (type, wing span code)
- airlines (codes, membership in alliances)
- destinations (airport codes, countries)

As Frankfurt Airport was taken as the sample airport for initial application of the research topic, a summary of most important data is provided below.

The airport complex consists of two passenger terminal buildings. As can be seen in Figure 32, there are five piers (A through E). For the time period looked at during research, the terminals have been connected via a people mover ('Sky Line') and via shuttle busses. A total of 153 gates have been used. Remote aircraft stands have not

---

<sup>66</sup> SQL: Structured Query Language is a standard for data retrieval and manipulation.

<sup>67</sup> AnySQL Maestro is a multi-purpose admin tool for database management, control and development. MySQL is a relational database management system. Both systems are available for free.

<sup>68</sup> The software taken for this was, Multi-Edit 2006, which is a commercial product.

<sup>69</sup> The data cleansing task took 2-3 weeks.

been looked at in detail.<sup>70</sup> For each of the gates its area in square meters and the maximum aircraft type it can serve (in terms of wing span code) have been gathered.

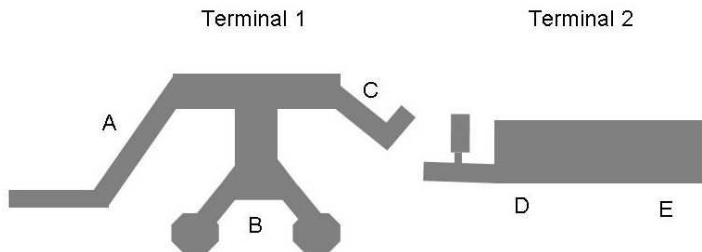


Figure 32: Basic terminal layout, Frankfurt Airport.

Seven retail areas have been defined (R1 through R7). They comprise the following gates:

| Retail Area | Gates     | Remarks   |
|-------------|-----------|---|
| R1          | A1 – A65  | EU, some non-EU   |
| R2          | B1– B19   | EU  |
| R3          | B20 – B59 | Transit, non-EU   |
| R4          | C1 – C22  | EU, non-EU  |
| R5          | D1 – D31  | EU, non-EU  |
| R6          | D40 – D54 | EU, non-EU, bus gates (spare) went into operation within research time window |
| R7          | E1 – E26  | EU, non-EU  |

Table 3: Basic definition of retail areas.

The retail areas are different in terms of their offering, their atmosphere, the passenger streams they serve, and of course in their location.

The data on aircraft basically consists of 219 different types of aircraft with their respective wing span code, ranging from 15 (smallest) to 1 (largest).

According to the data set, a number of 195 airlines served Frankfurt Airport in the time frame looked at. They served destinations with 1,597 different flights (flight numbers) during summer season and respectively 1,102 flights (flight numbers) during winter season.

The different airline alliances taken into consideration for the gate allocation task were: Star Alliance, SkyTeam, and Oneworld.

<sup>70</sup> This will be explained in Chapter 4.2.2 (standard assumptions).

Various destinations in 106 different countries have been identified and considered in their retail performance.

### 3.4.2 Flight schedules

The period of time that is investigated includes two flight plan seasons:

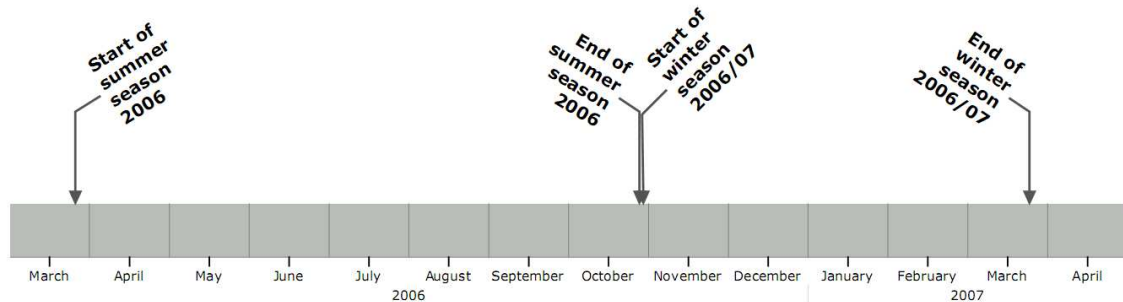


Figure 33: Timeline of flight plan seasons.

- summer season 2006 (2006-03-26 to 2006-10-28)
- winter season 2006/07 (2006-10-29 to 2007-03-24)

When talking about flight schedules it needs to be differentiated between actual flight data and the seasonal flight plan. The first reflects the traffic situation based on actual data (comprising scheduled time of departure (STD), last updated estimated time of departure (ETD) and the actual time of departure (ATD)). The seasonal flight plan e.g. lacks of course ETD and ATD. Furthermore, it has to be mentioned that after the slot return date airlines still return slots or make changes to their schedules. So the STD in the set of current flight data reflects the last change in STD since the seasonal flight plan has become valid. This may be as close as up to just prior to the day of operations.

In addition to that, it has to be taken into account that a seasonal flight plan usually consists of just a single (peak) week per flight plan season. Such a week is then taken as a master to allocate the flights of the following week (on a rolling day basis).

The original set of data made available for research consisted of more than two million data records with approximately fifty entries (fields) per record. So in total about one hundred million data items had to be pre-processed. After non-passenger flights (e.g. cargo, governmental, rescue) have been filtered out, and the above mentioned data cleansing tasks have been applied to the data set, still a remainder of 229,430 records (summer: 140,962; winter: 88,468) was left.



### **3.4.3 Retail sales data**

An improved gate allocation would not be possible without any information regarding the contribution of passengers on a flight towards retail sales.

As introduced above, the sales result depends on many different factors. A major (if not most important) role plays the number of passengers and their country of origin (the latter is assumed to be in close relation to a flight's destination).

Depending on the business model that an airport applies to retailing, it is the retailer or the airport operator who gathers data on individual purchases (usually at point of sale). In order to obtain information that relates a purchase to a flight, very often the boarding pass is scanned, or a combined data gathering consisting of passport scan and manual entry of flight number is conducted.

This information is crucial to identify any levers towards improved sales. Thus both, the airport operator and the retailer(s) should have a vast interest to make best possible use of this information. In some contracts between airport operator and retailer there can be found regulations regarding the shared use of such sales information. Such information may be at a very detailed level (e.g. on a transaction basis) or it may be on an aggregated level (both summed values and relative index values).

For the purpose of this research project such information has been provided by Fraport AG (the operator of Frankfurt Airport). They have obtained the information (on a more detailed level) from the major retailer on their premises (Gebr. Heinemann).

So, the retail data used is in form of a relative index (retail-worthiness), on the basis of average sales. The aggregation level is flight-specific. However, as this level of detail would obtain no permission to be published from the data owners, the information presented in this paper had to be aggregated on a country-specific level (see Table 4). Nevertheless, for the improvement process (algorithm and simulation software) the flight-specific level of detail has been applied.

| Pos. | Country            | Relative retail worthiness (%) |
|------|--------------------|--------------------------------|
| 1    | Korea, Republic of | 486.5                          |
| 2    | Kazakhstan         | 428.7                          |
| 3    | Russia             | 393.0                          |
| 4    | Uzbekistan         | 381.0                          |
| 5    | Nigeria            | 299.9                          |
| 6    | Belarus            | 299.5                          |
| 7    | Azerbaijan         | 280.5                          |
| 8    | Ukraine            | 275.1                          |
| 9    | Libya              | 258.4                          |
| 10   | Iran               | 257.5                          |
| 11   | Japan              | 255.1                          |
| 12   | Taiwan             | 253.5                          |
| 13   | China              | 251.8                          |
| 14   | Vietnam            | 245.3                          |
| 15   | Serbia             | 244.5                          |
| 16   | Turkmenistan       | 239.2                          |
| 17   | Algeria            | 222.4                          |
| 18   | Seychelles         | 201.9                          |
| 19   | Moldova            | 195.8                          |
| 20   | Tuvalu             | 190.9                          |
| 21   | Ethiopia           | 190.8                          |
| 22   | Romania            | 188.9                          |
| 23   | Iraq               | 184.9                          |
| 24   | Latvia             | 177.6                          |
| 25   | Brazil             | 173.4                          |

| Pos. | Country           | Relative retail worthiness (%) |
|------|-------------------|--------------------------------|
| 63   | Sri Lanka         | 101,9                          |
| 64   | Argentina         | 101.2                          |
| 65   | Grenada           | 100.8                          |
| 66   | Canada            | 100.3                          |
| 67   | Cyprus            | 99.0                           |
| 68   | Trinidad & Tobago | 97.0                           |
| 93   | Luxembourg        | 45.3                           |
| 94   | France            | 44.1                           |
| 95   | Pakistan          | 42.2                           |
| 96   | Spain             | 39.5                           |
| 97   | Italy             | 36.6                           |
| 98   | Greenland         | 34.3                           |
| 99   | Belgium           | 33.7                           |
| 100  | Netherlands       | 31.0                           |
| 101  | Germany           | 30.1                           |
| 101  | Kosovo            | 29.2                           |

**Table 4: Retail-worthiness of countries (selection) at Frankfurt Airport in 2007.**

An initial analysis of the data shows that occasionally there is a considerable bandwidth in the flights' individual retail-worthiness to the same country of destination. For example, a flight to New York may be at 180% of average retail sales, whereas a flight to Dallas achieves a rating of e.g. 300%. As mentioned above, the data provided contains information on a flight-specific level. Country-specific data is simply the weighted average of all flights to that country.

However, there is a drawback regarding the data provided: Due to the fact that the time frame is not the same as used with the operational (actual and seasonal) flight data, and the absolute retail figures are derived from annual reports, instead of directly from the

same data source of retail-worthiness figures, a comparison has to be drawn with care. So, an absolute statement regarding the actual retail results in respect to operations cannot be drawn. This needs to be taken into account in the interpretation of the results.

Nevertheless, for the purpose of research the data provided is a test environment, which is still close to real conditions. It may be due to the different time frame (retail data is from calendar year 2007), and certainly as well due to data inconsistencies that for some of the flights in the operational data no counterpart in the retail data could be identified. In such a case (within the data cleansing process) flights to the same destination airport have been taken to determine the retail-worthiness of the flight in question.

In case no flight with the same destination could be found, the country-specific value was applied. There has been no case of a flight (not included in retail data) to a destination in a country for which no retail-worthiness was known. Within the data cleansing process much effort had to be put into the correct representation of countries, because much inconsistency on a logical basis was determined (e.g. different representation of former Soviet Union countries in retail data compared to operational data).

However, the retail data provided allows further use in analysis and simulation. Furthermore, the anonymous nature of the information provided should encourage the application at different airports.

As introduced with the conceptual research model in Chapter 3.3, the rule set plays a vital rule for the overall result. The next chapter describes this in more detail.

#### **3.4.4 Rule set**

Very often business rules are not explicitly defined or written down somewhere. For example, many shops prepare for passengers by observing the information provided on public flight displays. The preparation may be as simple as to re-arrange shelves depending on the next major flight to be expected. Slightly different, in operations, business rules are often stored in IT systems and / or formally noted in operational directives (or similar). The better these two sets of business rules are harmonized with each other, the smoother and more successful business will be conducted. The planning task will try to identify competing rules and prioritize their application. Rules may also be complementary to each other (e.g. successful application of rule 1 will allow rule 2 to produce better results). The core business of a process usually follows rules that are indifferent to those of another processes' core business.



Sales for any single flight ( $S_{FLIGHT}$ ) is the product of the factors above, where  $S_{DF}$  and  $F_{DFR}$  in combination describe the average retail spending<sup>71</sup>, but still regardless of the flight number and the location where the flight is allocated at. This is then coped for by  $F_{FLIGHT}$  and  $F_{AREA}$ . As also those values have been defined and derived as *factors*, sales can be described as:

$$S_{FLIGHT} = P S_{DF} F_{DFR} F_{FLIGHT} F_{AREA} \quad (EQ.1)$$

Usually, there will be one or more flights to be allocated at a specific time. For this reason a day has been divided into 288 time intervals each representing 5 minutes of time. The number of flights ( $nF$ ) per interval is basically determined by the flight schedule for that day. Therefore, sales for a single time interval would be:

$$S_{interval} = \sum_{i=1}^{nF} (P S_{DF} F_{DFR} F_{FLIGHT} F_{AREA})_i \quad (EQ.2)$$

The gate allocation algorithm's task would be to maximize sales under the constraints of a specific scenario. Therefore, the sales result of an improved time interval is:

$$S_{interval(max)} = Maximize \left( \sum_{i=1}^{nF} (P S_{DF} F_{DFR} F_{FLIGHT} F_{AREA})_i \right) \quad (EQ.3)$$

Consequently, a day's improved retail sales figure may be expressed as:

$$S_{day(max)} = \sum_{t=1}^{288} \left( Maximize \left( \sum_{i=1}^{nF} (P S_{DF} F_{DFR} F_{FLIGHT} F_{AREA})_i \right)_t \right) \quad (EQ.4)$$

---

<sup>71</sup> In consequence, the as long as product of  $S_{DF}$  and  $F_{DFR}$  represent an airport's figure for passengers average retail spending, the individual factors are less important. However, they may be used, in case only duty free sales data would be available.

Applied to the period of time that the research project investigates, where  $Dstart$  = start date, i.e. 2006-03-26, and  $Dend$  = end date, i.e. 2007-03-27, the sales figure ( $S_2$ ) as expressed in the conceptual research model would then be:

$$S_2 = \sum_{d=Dstart}^{Dend} \left( \sum_{t=1}^{288} \left( \text{Maximize} \left( \sum_{i=1}^{nF} (P S_{DF} F_{DFR} F_{FLIGHT} F_{AREA})_i \right) \right)_t \right)_d \quad (\text{EQ.5})$$

The conceptual research model claims in (H.1) that  $S_2 > S_1$ , where  $S_1$  represents the sales result based on actual allocation (i.e. not retail-focussed). This leads to the quantitative research model:

$$\begin{aligned} & \sum_{d=Dstart}^{Dend} \left( \sum_{t=1}^{288} \left( \text{Maximize} \left( \sum_{i=1}^{nF} (P S_{DF} F_{DFR} F_{FLIGHT} F_{AREA})_i \right) \right)_t \right)_d \\ & > \sum_{d=Dstart}^{Dend} \left( \sum_{t=1}^{288} \left( \text{Actual} \left( \sum_{i=1}^{nF} (P S_{DF} F_{DFR} F_{FLIGHT} F_{AREA})_i \right) \right)_t \right)_d \end{aligned} \quad (\text{EQ.6})$$

Finally, it has to be mentioned that there may be potential inter-dependencies between  $F_{FLIGHT}$  and  $F_{AREA}$ . For example, may a flight with a low  $F_{FLIGHT}$  be ‘developed’ towards a higher  $F_{FLIGHT}$  when allocated in a retail area with a high  $F_{AREA}$  (nothing else changed)? And may a retail area with low  $F_{AREA}$  be ‘developed’ towards an increased  $F_{AREA}$  when only flights with high  $F_{FLIGHT}$  would be allocated to its gates (nothing else changed)? Such inter-dependencies may exist, but are not further investigated in this thesis.

[ALIGNMENT ASPECT]

*It is tried to model the real business world by identification of relevant data and thus enabling any potential IT solution to produce more meaningful output compared to a situation with pure artificial data (compare, Figure 10, (3) and (5)). The quantification of the conceptual research model supports determination of retail sales, and thereby supports performance measurement (compare, Figure 10, (6)).*

The next chapters describe the approach to determine an improved  $S_2$  as derived from the research model.

### 3.6 Approach to determine improved solutions

As discovered above, the model aims to maximize the retail result of flights in a specific time interval. Given a certain number of flights in each interval<sup>72</sup> and a maximum number of known gates, the improvement task is to find the best possible (and valid) combination.

#### 3.6.1 Type of problem and mathematical considerations

In addition to the general problem description in Chapter 2.3 the following helps to better understand the specific challenge within this paper. The task is to solve a combinatorial optimization problem within discrete mathematics. The following may describe the dimensions of the possible solution space and suggests a way to reduce solution space considerably.

Usually, there are up to 11 flights scheduled for departure within a single time interval. Those flights may be allocated to any of the possible 153 gates.

So, at first sight there may be as many solutions as possible permutations. At second sight, it has to be clarified that a flight will only be allocated once (and only once) at one of the 153 gates within a specific time interval. In addition, the sequence of the elements does not matter. This defines a sub-set of permutations, called combinations. Of course, depending on the period of time that a flight remains at the gate (for turnaround and / or boarding) such a gate would not be available to other flights in consecutive time intervals until the flight left the gate (and a potential buffer time would have elapsed).

Consequently, the number of 153 gates will decrease with more and more flights already being allocated. In order to cope with business requirements, many of the constraints (like aircraft size vs. gate size, alliance membership) will further decrease the possible number of gates. Finally, a certain number of '*eligible gates*' will remain as candidates for allocation.<sup>73</sup>

---

<sup>72</sup> Deviation from the (reference) flight schedule is to be kept to a minimum.

<sup>73</sup> This is similar to the approach of Murty et al. (2008).

However, the solution space is still very large for computational solving, because the number of potential solutions to be looked at in terms of their retail result seems to be calculated according to the general formula:

$${}_nC_k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Where:

- C** is the number of combinations from a set
- n** is the number of possible gate-flight allocations to choose from (i.e. the product of: for each flight all possible gates)
- k** is the number of gates to be chosen (i.e. number of flights that need to be allocated within a specific time interval)

As the definition of combinatorial problems tends to mislead, if not too familiar with it, a simple (often-used) example will be transferred to the gate allocation problem.

The example is the calculation of all possible combinations of a five-card hand taken from an Anglo-American style fifty-two card-deck.

The number of combinations would be:

$$C = \binom{52}{5} = \frac{52!}{5!(52-5)!} = 2,598,960.$$

In order to apply this to the gate allocation problem, the figure of 52 needs to be broken down into its factors of 13 cards per suit. Figure 34 shows a first translation into the research problem's context:



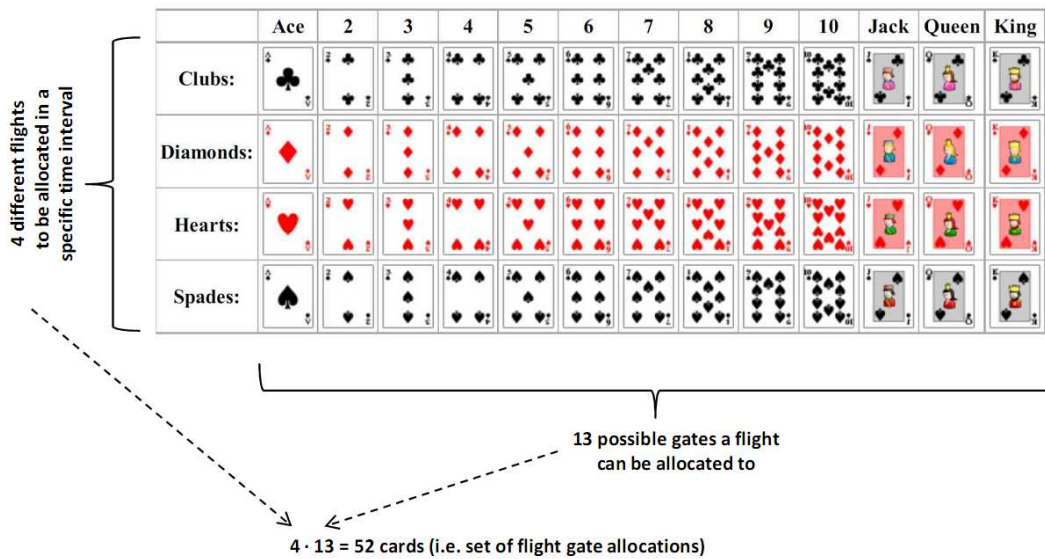


Figure 34: Card-deck example transferred to gate allocation context (1).

For the combinatorial problem this would lead to a value for n of:

$$n = 11 \text{ flights} \cdot 153 \text{ gates} = 1683.$$

So, for a single time interval, in worst case (n= 1683, k = 11) the number of combinations would be:

$$C = \frac{(1683)!}{11!(1683-11)!} = 3.9274226487240218209005140924565e + 4700.$$

Actual flight data more often shows a case like this (n ≈ 9·65 = 585, k = 9):

$$C = \frac{585!}{9!(585-9)!} = 20,786,604,884,463,688,985.$$

Assuming that only half the day so many flights (9 per interval) would need to be allocated, there would still be 144 intervals, totalling

$$2,993,271,103,362,771,213,840 \text{ (combinations) trial allocations per day.}$$

Keeping in mind that 364 days within the two flight plan seasons are to be looked at, this approach does not promise to be feasible in terms of computing power available.

However, most importantly, this approach would not be correct and valid.

The abovementioned transfer to the research context misleads in a certain point: a valid 5-card hand may very well be 5 to 9 of clubs:

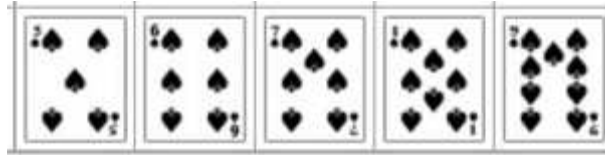


Figure 35: Card-deck example transferred to gate allocation context (2).

However, in research context this would mean that a possible flight-gate-combination would be something like flight #28 at gates B4, B5, B6, B7 and B8 at the same time. This of course is not possible (nor is it allowed or makes sense).

For the correct representation of the problem the combinatorial rules have to be applied. The *rule of product* states that for  $x$  possibilities of performing A and for  $y$  possibilities of performing B there are  $x \cdot y$  possibilities of performing both under the condition that A and B can be performed at the same time (so that they are not mutually exclusive, which would mean either A or B can be performed, but not both<sup>74</sup>).

Applied to the research context this would mean:

For each flight to be allocated, exactly one gate (at a time) out of 153 gates needs to be chosen. With 11 flights and 153 gates this results into

$$153^{11} = 1,075,488,420,943,298,174,695,497 \text{ combinations.}$$

With 9 flights out of 65 gates, it still leads to

$$65^9 = 20,711,912,837,890,625 \text{ combinations.}$$

Although this is already far less than before (approx. to the factor of 1,000), it is still too huge for ordinary computation power.

For this reason, the possible solutions space had to be reduced.

The goal is to find the optimal valid (i.e. under the constraints given) combination of flights and gates. Here '*optimal*' means to achieve a high retail result. As introduced above, the latter depends on the retail area, which a flight is allocated to (and thus only indirectly on the gate itself). This simple change promises to reduce solution space to a large extent. As there are only seven possible retail areas (R1-R7), the corresponding number of combinations per time interval would be:

<sup>74</sup> In such a case the rule of sum would apply.

| Number of flights | Number of combinations |               |
|-------------------|------------------------|---------------|
| 11                | $7^{11}$               | 1,977,326,743 |
| 10                | $7^{10}$               | 282,475,249   |
| 9                 | $7^9$                  | 40,353,607    |
| 8                 | $7^8$                  | 5,764,801     |
| 7                 | $7^7$                  | 823,543       |
| 6                 | $7^6$                  | 117,649       |
| 5                 | $7^5$                  | 16,807        |
| 4                 | $7^4$                  | 2,401         |
| 3                 | $7^3$                  | 343           |
| 2                 | $7^2$                  | 49            |
| 1                 | $7^1$                  | 7             |

**Table 5: Number of potential combinations in (initially) reduced solution space.**

A day with just 144 intervals with 9 flights would then require 5,810,919,408 combinations (and 2,115,174,664,512 for all 364 days<sup>75</sup>) to be generated, tested for validity and calculated for revenue. However, despite this enormous reduction in the number of possible solutions, it still constitutes a burden to compute them. In addition to that, the objective to determine variables that influence the result requires different scenarios to be tested. In consequence, the computing effort would need to be multiplied by the number of scenarios.

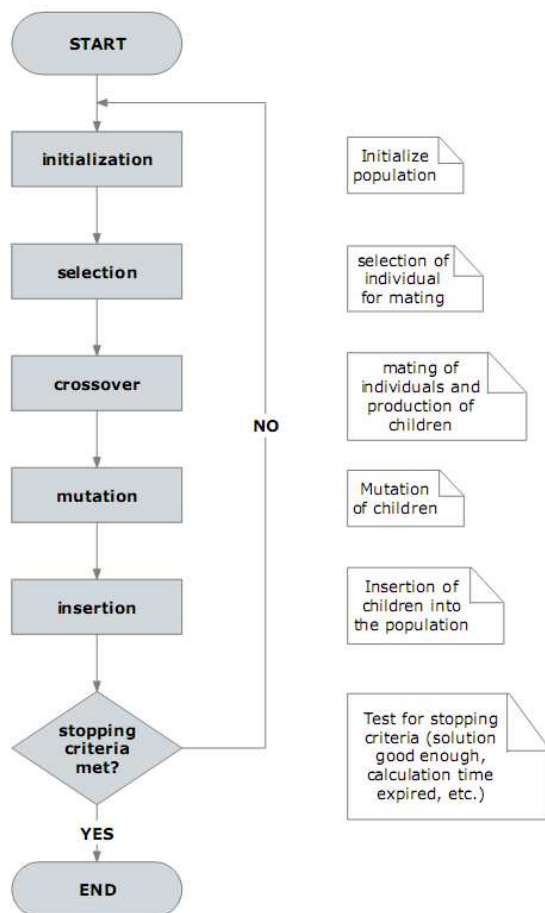
The means chosen to overcome this challenge are discussed in the next two chapters.

---

<sup>75</sup> Decimal notation has been chosen where possible to avoid ambiguity in naming convention (e.g. 1,240,565,629,343,040 in American system would be approx. 1.2 Quadrillion, in British system it would be approx. 1.2 Thousand Billion), and for better imagination compared to scientific notation. (Am: Thousand < Million < Billion < Trillion < Quadrillion < Quintillion <...; Br: Thousand < Million < Thousand Million < Billion < Thousand Billion < Trillion-...)

### 3.6.2 Heuristic approach versus deterministic approach

Previous research on the gate assignment problem state that good results (especially with a multi-objective function) have been achieved, using heuristics (e.g. Haghani and Cheng, 1998). So, at the beginning of the research project a heuristic approach has been followed, implementing the improvement part of retail sales by means of a genetic



algorithm (GA). In general, with such an approach relatively good solutions can be found in a short (or defined) period of time. However, the solutions found may not be the optimum result. The basic GA usually follows an approach as outlined in Figure 36. The solution to a problem is called a chromosome. Each chromosome has different genes. The value of a gene is called an allele. A population (i.e. a generation) consists of many individuals (chromosomes). The genetic operators (selection, crossover, insertion, mutation) generate (or breed) a new generation. The fitness of the generation usually is one of the stopping criteria for the algorithm. The coding of a chromosome is often represented by values of '0' and '1'.

Figure 36: Flowchart of a basic genetic algorithm.

However, as this would only describe the genotype, it is important to define a 'real world representation', called a phenotype. For example, the value (allele) of '1' on a certain gene of a chromosome may indicate that gate 'B46' is occupied. In order to implement such a GA in software, the respective procedures for the major steps as outlined in Figure 36 would need to be developed. Because of the complex validity checks of a solution and the revenue calculation, it was not feasible to use standard GA

---

packages. A trial has been conducted using a Microsoft Excel-based GA add-in<sup>76</sup>. The pre-processing and coding into the genotype has been done using self-developed software. Then an automated hand-over to an Excel spreadsheet has been initiated.<sup>77</sup> Within Excel the add-in and some self-written VBA<sup>78</sup>-code generated a possible solution by means of the GA. The result was then handed back to the custom software, which translated the genotype backwards into a phenotype. Finally, validity and revenue of the suggested solution have been determined. Unfortunately, the inter-programme communication by means of OLE (and also by means of a shared file) took far too much time to produce results. A complete development of a specialized GA package with specific genetic operators and rates for mutation and crossover would have had to be developed as no ready to use packages or suitable programming libraries could be found.<sup>79</sup> But as the objectives of this paper require a usable environment for scenario simulations, a more feasible way had to be looked for.

With further reduction of the solution space or with stepwise elimination of possible solutions that do not promise to be valid or to improve the so far best solutions, a deterministic approach was looked at.<sup>80</sup> The basic difference towards a heuristic approach is that each possible element in the solution space is considered. Thus at the end of such an algorithm the result can be described in definite terms.

Nevertheless, in order to minimize processing time, the algorithm uses an optimistic approach to determine the solution. Such an approach has not been seen before, but found useful in the problem's context. The next chapter describes this in detail.

---

<sup>76</sup> Product called: Evolver.

<sup>77</sup> Hand-over was realized by means of COM automation (OLE).

<sup>78</sup> VBA: Visual Basic for Applications is a Microsoft Office internal programming language.

<sup>79</sup> Note: There are both free and commercial GA packages available. But either the price was too high, the implementation turned out to be too complicated (given the time constraints) or they did not support the Windows programming environment (using DLLs).

<sup>80</sup> Similar to a branch-and-bound approach.

### 3.6.3 Deterministic algorithm for retail-oriented gate allocation

In reference to the gate assignment problem (GAP) classification within the literature review the proposed solution is characterized in Figure 37:

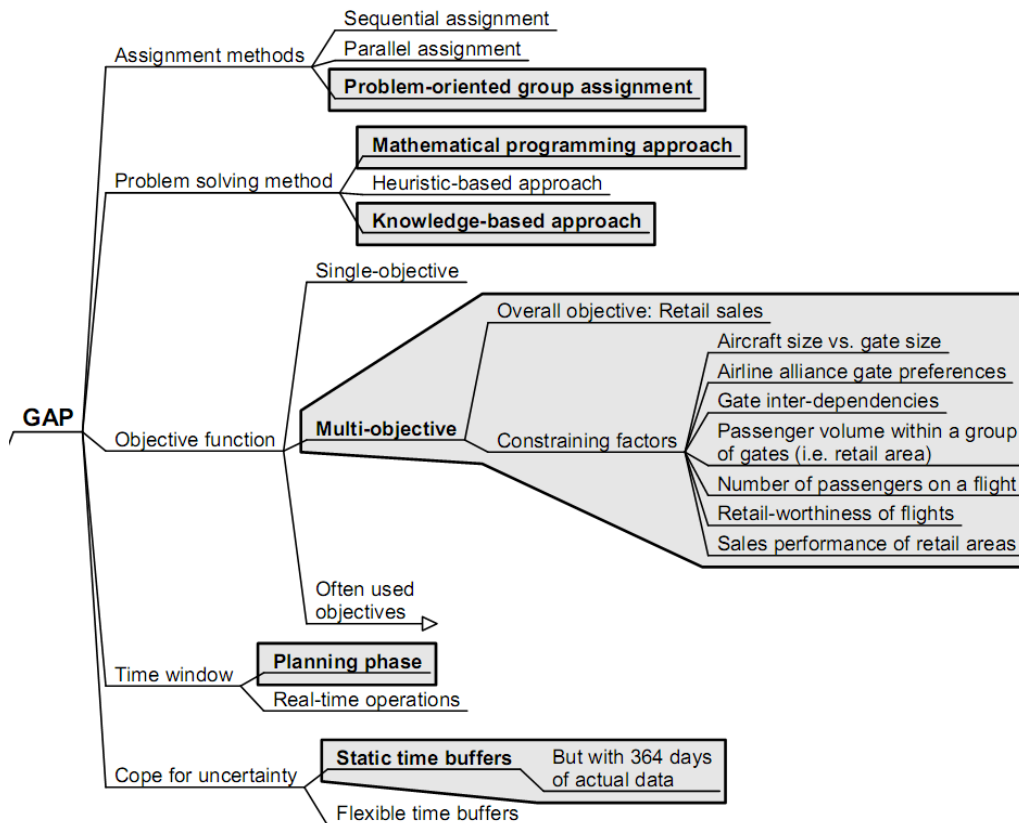


Figure 37: Proposed solution (framed branches) within GAP classification.

The problem-oriented group assignment method has been chosen, because firstly the sequential approach would not have left any room for improvement and no strict order (as defined by this approach) was given within each time interval. Secondly, the parallel assignment method would waste computing power (by unnecessarily expanding solution space) because of the problem’s nature. Trying to assign a flight at e.g. 11:55 and at 16:20 concurrently is not necessary. So, reducing the parallelism to a 5 minute interval, and working on those intervals in a time-wise order, classify the approach being a so-called ‘problem-oriented group assignment method’.

Regarding the problem solving method a heuristic approach would have been as valid as a deterministic approach. For reasons as outlined in the previous chapter, basically a deterministic (mathematical programming) approach has been followed. In order to produce ‘closer-to-real-life’ solutions, expert knowledge in form of many constraining

---

factors have been incorporated in the solution finding process. This is referred to as ‘knowledge-based approach’. A good example of such expert knowledge is the ‘gate inter-dependency’. This simply means that a certain gate is only usable when a defined dependent gate (usually in direct neighbourhood) is not occupied. Another example from within the objective function constraining factors is the retail-worthiness of a flight (as already explained in Chapter 3.4.3 and introduced as  $F_{FLIGHT}$  in Chapter 3.5).

Additional objectives might have been considered (e.g. airport business lounges), but at the same time would have reduced solution space. Therefore, only the most influential ones in respect to retail sales and gate allocation have been considered. Customer (i.e. airline) wishes have been summarized in the alliance rule (Chapter 3.4.1, pre-last paragraph).

Furthermore, it needs to be mentioned that because of the fact that a whole year of actual flight data has been used, also any charter flight in that time has been considered in the process.

The algorithm addresses solution finding for exactly one day. In order to calculate many days, it simply needs to be run once per improvement candidate (day). The basic assumption is that there is a fixed flight schedule, which is not altered prior to the allocation process. In a case within the allocation process, where flights cannot be allocated as requested, they will obtain an ETD, which is later compared to the STD.

Flights that could not be allocated during a day at all, will lead to a corresponding result of the algorithm (‘no successful allocation possible’). This would indicate that the flight schedule may be too tight, too few gates were available or too many restrictions have been put into the rule-set. In such a case allocation will abort and a re-design of the flight schedule is suggested.<sup>81</sup>

---

<sup>81</sup> In the case of Frankfurt Airport’s traffic this did not occur.

The algorithm basically consists of three phases and applies an enumeration type of approach:

Phase one does pre-processing like:<sup>82</sup>

- *Determination of flights* that need to be allocated in the time interval to be improved. This comprises flights scheduled in the current time interval and flights that may not have been allocated in the previous interval(s).
- *Determination of those gates*, which are *still available* in the current time interval. Due to the nature of the algorithm (time-wise strictly moving forward, never backwards) a gate will always be available long enough once it has been detected available in an interval. A gate can be unavailable for many reasons. This will be discussed in Chapter 4.2.2 (standard assumptions, rule set).
- *Determination of those gates*, which are *eligible* for each of the flights. A gate can turn out to be not eligible for many reasons. This too will be discussed in Chapter 4.2.2.
- *Determination (i.e. calculation) of revenue* for any possible flight–gate–combination (and for flight–retail area–combination).

Phase two conducts the core combinatorial task, which consists of:

- Production of solution candidates on a *retail area basis*. As mentioned before, production of solution candidates on a retail area basis considerably reduces solution space, but at the same time introduces a first element of optimism to the algorithm. This is because there may be an available gate in a retail area, but as more than one flight may find a certain retail area to be the most suitable one, there may be not enough free gates in that retail area for all flights to be allocated.
- Storage of a certain number of the best solution candidates in a solution stack. The stack has been sized 5000 elements, which means that in descending order the best 5000 solution candidates will be tested for allocation on a *gate basis* later. The number of possible elements in the solution stack basically describes the level of optimism to find a valid solution. The higher the number the less optimistic (i.e. more pessimistic) solution finding is appraised to be. For example, a stack size of one element (most optimistic) would mean that only the

---

<sup>82</sup> Determination of gates in phase one is similar to Murty et al. (2008).



best solution would be stored and later on (in phase three) tested for allocation on a gate basis.

- Limitation of processing time. The combinatorial task was constraint to 60 seconds per interval. This means that no more combinations will be produced after that time will have elapsed. This allows running the algorithm unattended (i.e. not risking to wait endlessly in case of a so-called combinatorial explosion when for example 20 instead of 11 flights would need allocation in a specific time interval). This introduces another element of optimism to the algorithm.

Phase three does post-processing like:

- To try an initial *gate* allocation for the suggested optimum combination(s) of flight and retail area. In case no combination of the solution stack could be allocated, a second iteration without the constraint of alliance membership<sup>83</sup> is tried.<sup>84</sup>
- Block any gates in a retail area that would be too crowded by passengers in case additional flights would be allocated there (a standard value of 1.5 square meters per passenger in gate hold room has been applied<sup>85</sup>).

---

<sup>83</sup> This rule turns an allocation trial invalid in case a flight of a member airline of an alliance would be allocated to gates not associated with that alliance. However, in simulation runs this situation did occur in one scenario only.

<sup>84</sup> Not modelled into the Nassi-Schneiderman-diagram for reasons of overview and simplicity.

<sup>85</sup> IATA recommendations: 0.6 to 1.4 m<sup>2</sup> for gate hold rooms and 1.0 to 2.7 m<sup>2</sup> for long term waiting space (Kazda and Caves, 2000, p.253).

Figure 38 shows the above-described basic algorithm in Nassi-Schneiderman notation:

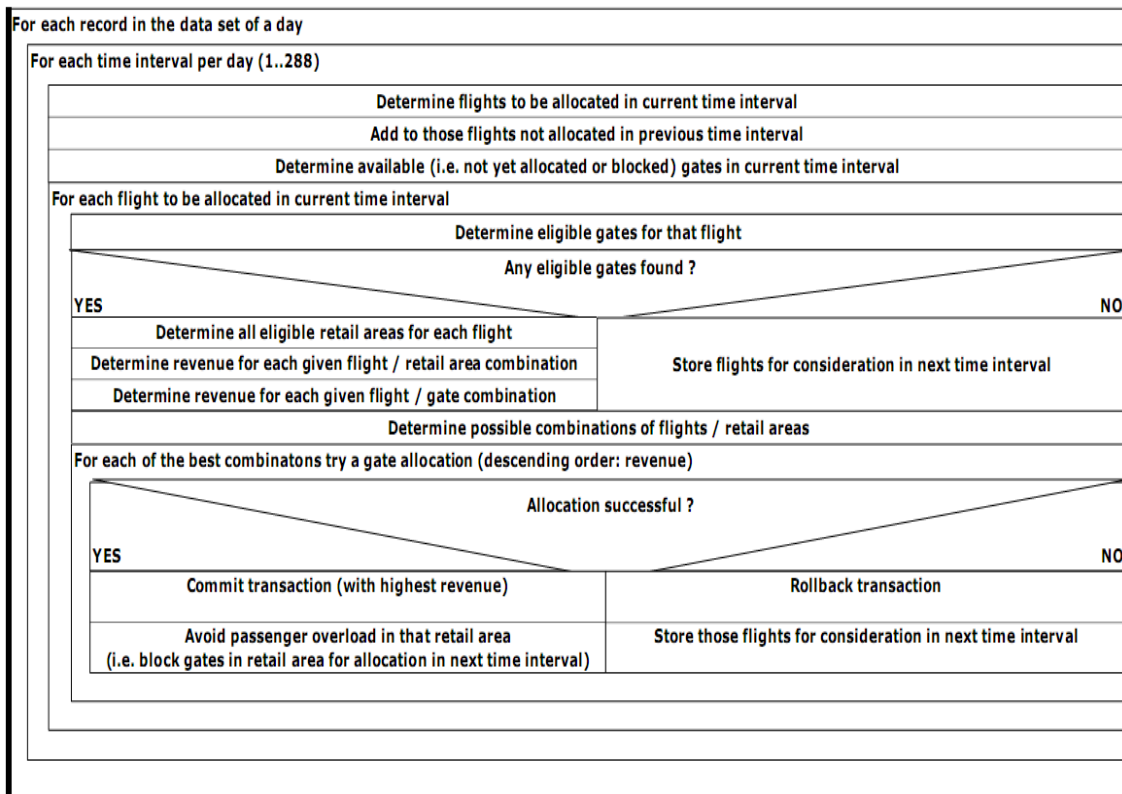


Figure 38: Algorithm for gate allocation, top level.

In order to appraise the quality of the solution, an internal variable sums up the difference between the retail results of the combination allocated and the best combination of the solution stack. This sort of opportunity cost indicates how much additional money could have been earned in case enough gates would have been available (or fewer constraints would have been applied). Nevertheless, under the conditions given always the optimum (feasible) combination is allocated.

The modular structure of the algorithm allows for a different implementation of the combinatorial part. For example, it would also be possible to fill the solution stack with combinations, resulting from a heuristic approach. Another possibility would be to fill it on a gate basis (instead of retail area basis) with a then possible reduction of solution stack to one element only.

The implementation of the algorithm forms part of the simulation environment, which will be introduced in Chapter 3.7. Being an important element of this research work, the source code, is provided in Appendix A.

Due to the fact that the source code comprises several thousand lines, it may be difficult to browse through in this paper version of the thesis.

Hence, for convenient access, Table 6 guides the reader towards the corresponding sections of the algorithm’s core parts. The names of functions and procedures are similar to those in the diagram of Figure 38.

| <b>Function or procedure</b>                                      | <b>page</b> |
|---|-------------|
| OPTI_Run()  | 242         |
| OPTI_FindSolution()   | 243         |
| OPTI_Determine_FlightsToBeAllocatedInTimeInterval()               | 261         |
| OPTI_Determine_AvailableGatesInInterval()                         | 260         |
| OPTI_DependendGateIsFree()  | 267         |
| OPTI_Determine_EligibleGatesForFlight()                           | 261         |
| OPTI_IsValidGate()  | 261         |
| OPTI_Determine_EligibleRetailAreasForFlight()                     | 263         |
| OPTI_Determine_RevenueForFlightInSpecificRetailArea()             | 263         |
| OPTI_Determine_RevenueForFlightAtSpecificGates()                  | 264         |
| OPTI_Determine_MaxTheoRevenue()                                   | 276         |
| OPTI_CombiTwoElements()                     [with recursive call] | 250         |
| OPTI_IsValidRACombi()   | 268         |
| OPTI_Avoid_RetailArea_PAX_OverLoad()                              | 275         |
| OPTI_BlockGatesInRetailAreaInTimeInterval()                       | 276         |

**Table 6: Directory to source code of algorithm.**

In order to explain the process of combination in the research context, the following will serve as an exemplary run:

- There are 7 retail areas given (#1 to #7).
- There are 3 flights to be allocated (#21, #28, #32).
- Each valid combination results in sales (intersection of row and column in the tables below).

As learned in the chapter about the mathematical background of the problem, the number of combinations would be  $3^7 = 343$ . So, a total of 343 possible combinations would enter the solution stack. But due to the nature of the research problem, (and as a further means of reduction in solution space) only the eligible (instead of the available) retail areas will be allowed to enter the combinatorial solution finding process. Thus the possible number of solutions will be less than that of a standard combinatorial problem.

|        |    | ELIGIBLE RETAIL AREAS |     |     |            |           |     |     |
|--------|----|-----------------------|-----|-----|------------|-----------|-----|-----|
|        |    | 1                     | 2   | 3   | 4          | 5         | 6   | 7   |
| FLIGHT | 21 | ---                   | --- | --- | ---        | <b>55</b> | 27  | 37  |
|        | 28 | 19                    | 17  | 28  | <b>101</b> | 7         | 55  | 66  |
|        | 32 | 7                     | 5   | 10  | <b>120</b> | ---       | --- | --- |

Table 7: Combinatorial example (1): reduced solution space.

According to the rule of product in the situation as given in Table 7 there will be

$$3 \cdot 7 \cdot 4 = 84 \text{ combinations.}$$

The bold figures indicate the retail area with the highest results for each flight. In case there were enough free gates in these retail areas, the combination of

flight #21 in retail area #5 ; flight #28 in retail area #4 ; flight #32 in retail area #4

would produce the highest revenue.

The combinatorial search would need to scan through the complete solution space as indicated in Table 8.<sup>86</sup>

|       |       |       |              |     |       |       |     |       |
|-------|-------|-------|--------------|-----|-------|-------|-----|-------|
| 5-1-1 | 5-2-1 | 5-3-1 | 5-4-1        | ... | 5-7-1 | 6-1-1 | ... | 7-7-1 |
| 5-1-2 | 5-2-2 | 5-3-2 | 5-4-2        | ... | 5-7-2 | 6-1-2 | ... | 7-7-2 |
| 5-1-3 | 5-2-3 | 5-3-3 | 5-4-3        | ... | 5-7-3 | 6-1-3 | ... | 7-7-3 |
| 5-1-4 | 5-2-4 | 5-3-4 | <b>5-4-4</b> | ... | 5-7-4 | 6-1-4 | ... | 7-7-4 |

Table 8: Combinatorial example (2): retail area combinations.

<sup>86</sup> Note: in a recursive implementation, the solution generation would start at the end with 7-7-4. From that point all prior recursive calls will be terminated in reverse order of calling, and thus building the individual solutions (see also below).

The grey highlighted combinations from the above table are shown with their respective sales result in Table 9.

|                  |                  |     |                   |                  |     |                   |
|------------------|------------------|-----|-------------------|------------------|-----|-------------------|
| 5-1-1            | 5-2-1            |     | 5-7-1             | 6-1-1            |     | 7-7-1             |
| $55+19+7=$<br>81 | $55+17+7=$<br>79 | ... | $55+66+7=$<br>128 | $27+19+7=$<br>53 | ... | $37+66+7=$<br>110 |

**Table 9: Combinatorial example (3): resulting retail sales.**

As mentioned before, the maximum retail sales figure would be produced with the retail area combination as shown in Table 10.

|     |     |     |                      |     |
|-----|-----|-----|----------------------|-----|
|     |     |     | 5-4-4                |     |
| ... | ... | ... | $55+101+120=$<br>276 | ... |

**Table 10: Combinatorial example (4): optimum combination.**

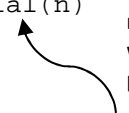
Given a solution stack size of only 10 elements, the 10 best combinations (in terms of highest sales result) would then be returned to phase three of the algorithm as described above. The reason is found in the way eligible retail areas are determined. An area may only be eligible if at least one gate is available in that retail area (and the other constraints are met). However, in case exactly one gate is available, but two flights would need to be allocated to it, a conflict arises. As mentioned before, this trade-off for reduction of solution space is addressed by the solution stack. Additionally, phase two of the algorithm addresses this issue in a way that a combination is only entered into the solution stack, if there are as many free gates in a retail area as there are flights that require gates in this same retail area.

As learned from the above example, it would be possible to run in loops through the solution space. However, as the degree of nesting differs, a varying number of loop levels (and iterations at each level) would need to be implemented in software. A more elegant way to describe and implement the above search of the solution space is by application of a recursive approach.

The same way a factorial number can be calculated both ways, the phase two of the algorithm is implemented in a recursive manner.

To outline the two different approaches possible, Table 11 provides an example for the calculation of the factorial of 3, where  $3! = 3 \cdot 2 \cdot 1 = 6$ .

A recursive approach<sup>87</sup> would define it as 3 times the factorial of 2. ( $3! = 3 \cdot 2!$ )

| Iterative approach  | Recursive approach   |
|---|--|
| <code>FACTORIAL (3) = 3 · 2 · 1</code>  | <code>FACTORIAL (3) = 3 · FACTORIAL (2)</code>   |
| <pre>n := 3 f := factorial(n)  function factorial(n)   t := n   for i := t-1 to 1 step -1     t := t * i   next i   return t end function</pre> | <pre>n := 3 f := factorial(n)  function factorial(n)   if n&lt;=1 then     return 1   else     return n * factorial(n-1)   end if end function</pre> <p style="text-align: right; margin-right: 50px;"><b>recursive call<br/>with updated<br/>parameter</b></p>  |

**Table 11: Pseudo-code for different implementations of the factorial function.**

Most important in definition and implementation of recursions is that the end of a recursion has to be defined and that it is detected prior to the next recursive call.

In the factorial example this would be the situation of  $1!=1$  and  $0!=1$ .

With the aforementioned in mind, Figure 39 presents a flow chart of the recursive implementation of phase two of the gate allocation algorithm. In that phase the reduced solution space is searched in a recursive (and partly optimistic) manner.

<sup>87</sup> An implemented recursion basically means that a function calls itself.

The initial call (to the START entry point) is conducted from within phase two of the gate allocation algorithm.

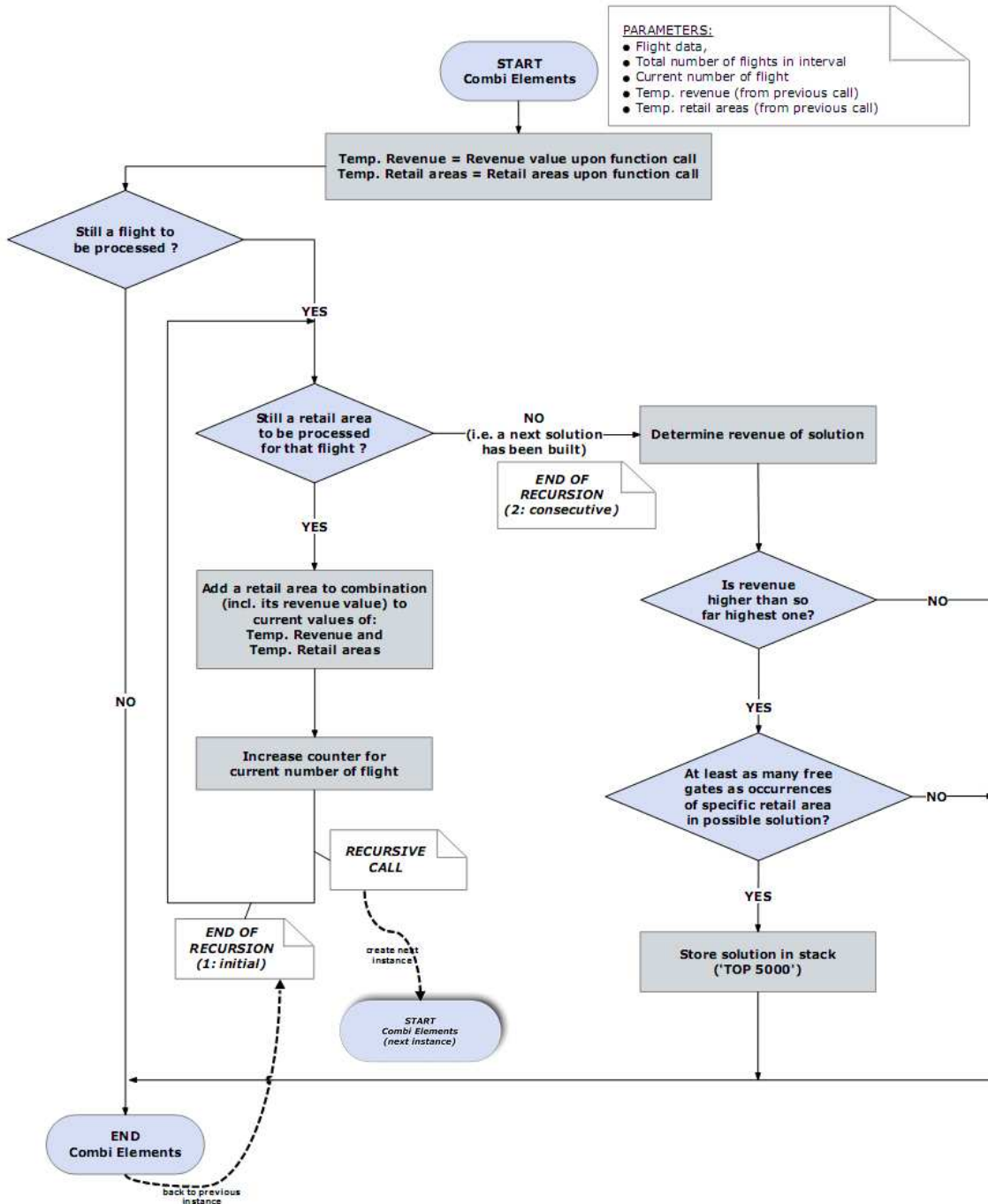


Figure 39: Algorithm to compose an allocation (recursive search of solution space).

After having returned from the initial recursive call (which will be the last one to close) the recursive implementation of the combinatorial search algorithm hands over control back to the calling point, which was in phase two of the gate allocation algorithm.

As defined, the algorithm performs recursive calls of itself until the last flight is reached (dotted lines in Figure 39). So a first retail area will be determined. Starting with the last flight, then a second retail area will be chosen for the pre-last flight and so on. Each time when stopping-criteria are met, a recursion ends and control jumps back one level. Within this level then the values are still the same compared to the moment in which control had been handed over to the level it just returns from. This way a solution builds up 'from the tail'. Solutions, which are both valid and 'fit' enough, will enter the solution stack. This stack is available to the calling function in phase two of the gate allocation algorithm.

With the implementation of the above it would become possible to run complete allocations with different settings. In order to detect potential sensitivity of certain parameters a systematic approach was necessary. Chapter 4.2 will discuss the different scenarios in detail.

#### *[ALIGNMENT ASPECT]*

*The previous chapters aimed to provide the foundation for an IT solution that is capable of producing desired output in an environment, which has as little requirements as possible regarding computing power in order to keep investment cost and operations cost low (compare, Figure 10, (4) and (5)).*

However, in order to be able to conduct simulation runs, an appropriate simulation environment needed to be developed. As this constitutes a major part of the research work undertaken, the following chapters specify the individual elements of the simulation environment, its implementation and general processing characteristics.



### 3.7 Simulation environment

As the term simulation is subject to misleading interpretation a brief outline of its meaning in the research context is given below.

The method of simulation (by means of the simulation environment) will be used to analyse the conceptual research model (using different scenarios for a sensitivity analysis). The latter is an abstract system subject to analysis. As a first step in this deductive approach the conceptual model was refined towards a quantitative model. In order to gain insight regarding that model, its individual elements will need to be analysed. Any knowledge gained from those individual aspects may help to explain the model itself.<sup>88</sup>

It is advantageous to use simulation in a situation where sole theoretical treatment of a question would lose transparency or is simply impractical.

This paper understands simulation as the numerical study of the quantitative research model. Visualization of simulation results is not a core part of the simulation environment. As most often used in an airport context<sup>89</sup> a discrete, event-driven simulation model is applied within this research.

The objective of the simulation environment is to represent the quantitative research model. It has to be able to generate output depending on different input parameters. It is aimed to use it as a kind of ‘research workbench’, and explicitly not to be a product of its own. More specific requirements are listed below:

- Ability to produce output in an acceptable amount of time on standard personal computer hardware.
- Ability to run different scenarios independently from each other on the same machine (or to split a scenario into several parts and merge the results after simulation runs have finished). In case of a multiple-core processor make use of all available cores for the simulation task.
- Allow for different data sources as input.
- Produce output that can be worked on with any standard software (e.g. Excel).
- Enable remote administration from anywhere in the world.
- Does not cost any money (except for own development effort) to operate.

---

<sup>88</sup> This in return would then be an inductive approach.

<sup>89</sup> Compare also Cheng (1998, p. 226-227).

With the requirements in mind, a general architecture of the ‘simulation workbench’ has been designed.

### 3.7.1 Components

The architecture follows the basic principle of Input→Processing→Output.

The input source can either be in form of data files or be manual input for information like dates of the simulation time frame (i.e. start date, end date).

Figure 40 shows the basic structure.<sup>90</sup>

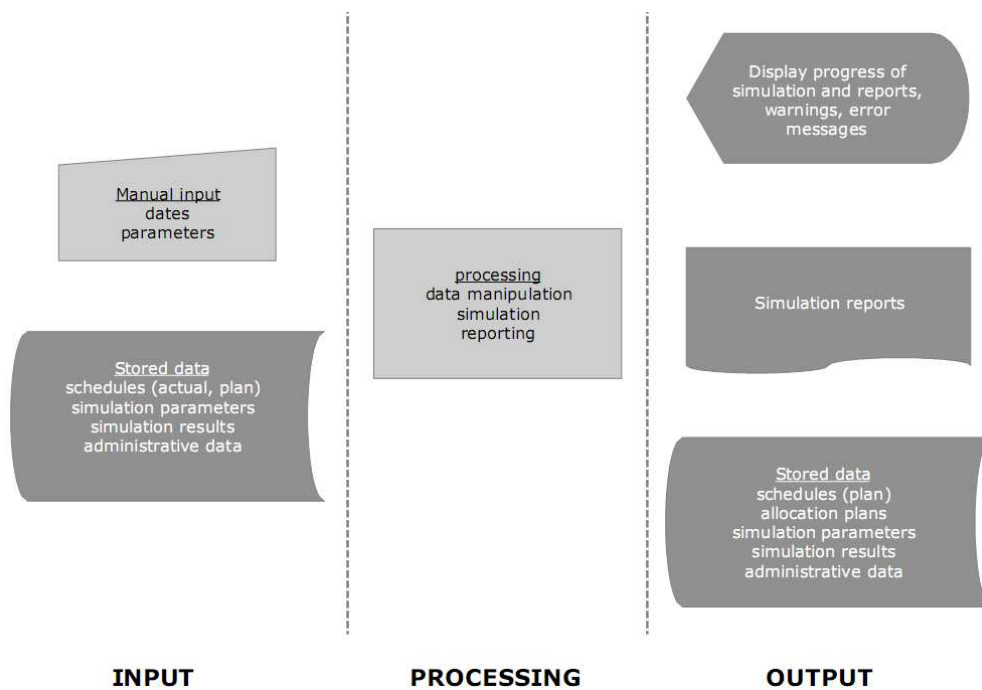


Figure 40: Basic architecture of simulation workbench.

Derived from the Airport Information Model (see Chapter 3.2.5), the simulation software needs to comprise those elements necessary to compute scenarios as described in the quantitative research model. In addition, there is data necessary for internal tasks of the software. These tasks include checks on files used, user-modes, or number of concurrently started instances of the software.

<sup>90</sup> A detailed description of the software architecture with all functions and procedures explained would be beyond the scope (and not in focus) of this paper. Nevertheless, the core part has been introduced in form of the gate allocation algorithm. The source code is provided in Appendix A.

So, basically there are three core categories of data the simulation workbench processes:

- reference data,
- working data,
- administrative data.

Additionally, in a fourth category there is various data generated for output (reports, messages on display). Figure 41 visualizes this in greater detail.

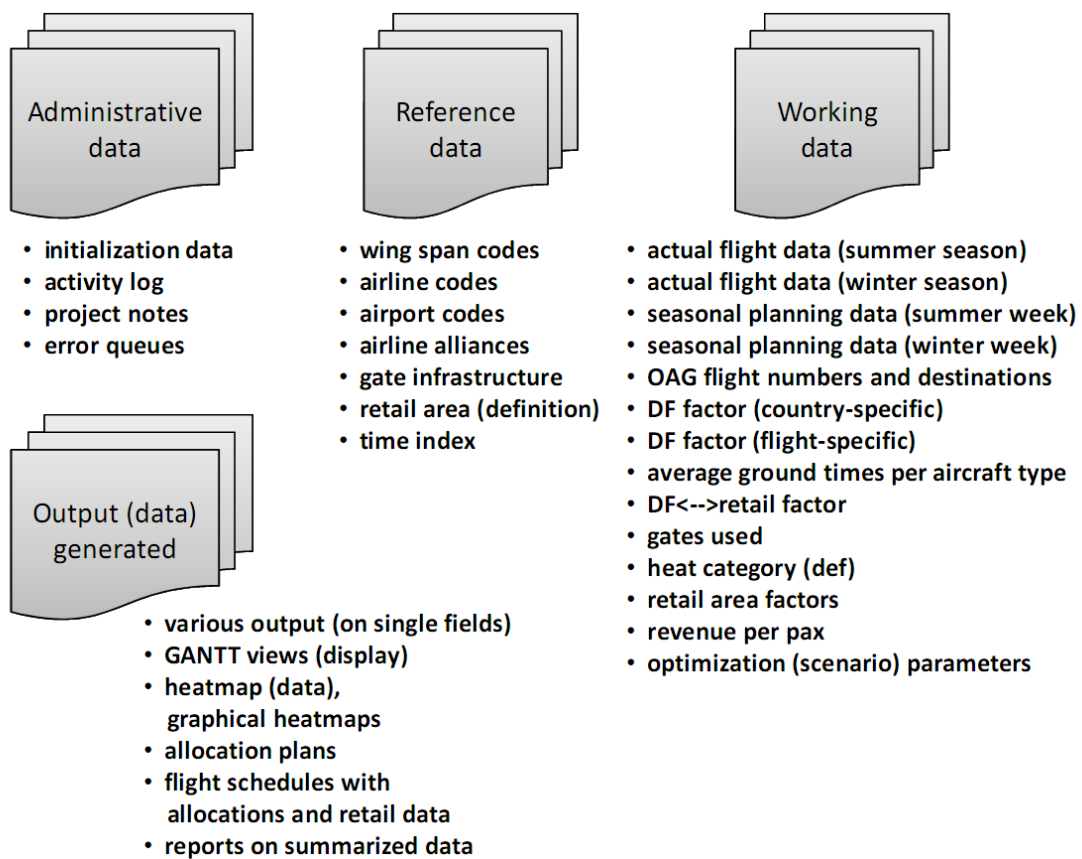


Figure 41: Data categories in simulation software, implemented as flat files.

In order to comply with most of the simulation workbenches' requirements storage of data has been implemented in form of flat text files. This allows for easy access directly by the software, and for easy generation of multiple instances with no additional administrative overhead and no additional run-time overhead.<sup>91</sup> However, in a professional production environment the use of database management systems is encouraged for reasons of internal data integrity and access security.

<sup>91</sup> And of course no additional cost is generated.

Further, the degree of normalization chosen is a compromise between highly normalized information entities (usually 3<sup>rd</sup> normal form) and explicit redundancy.<sup>92</sup> For example, in 3<sup>rd</sup> normal form within a data record of a specific flight the attribute (field) of ‘gate’ would need to reference to another entity (table) which contains all gates. Such an entry would be a reference pointer to the n<sup>th</sup> element in the ‘gate’-table. This usually ensures referential integrity. However, in the case of the simulation workbench, the software itself copes for that and allows for some degree of redundancy. In addition, the above makes it easier for third-party products to work with the output generated by the software.

As the basic architecture suggests, the data worked with is for input, processing and output. These tasks are usually accomplished by functions or procedures.<sup>93</sup>

In order to provide an overview of the basic functions (and procedures), Figure 42 shows the major functional elements of the software.

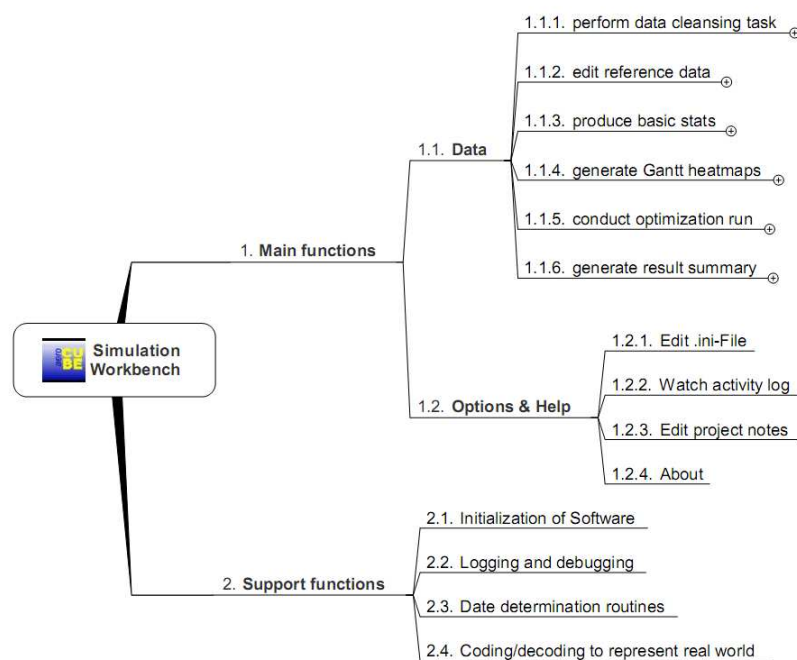


Figure 42: Function tree of simulation software, top level.

<sup>92</sup> Usually, in data warehousing (as part of business intelligence) there is a high level of explicit redundancy. This positively contributes to performance and transparency regarding the information stored in a data cube.

<sup>93</sup> The different terms ‘function’ and ‘procedure’ are used to indicate that a ‘function’ is expected to return a value whereas a ‘procedure’ does not. Nevertheless, depending on the scope and visibility of programme variables, a procedure may change globally defined variables and thus indirectly act as a function. On the other hand, a function may return a dummy value that is not considered any further in the programme (and thus acting like a procedure).

In general, it can be distinguished between functions<sup>94</sup> that are somehow interactive to the user of the workbench (main functions) and those that perform their work in the background (support functions). Within the main functions it can be further differentiated between those that serve the core data cleansing and simulation task, and those that help the user to plan tasks as well as to keep track on her/his activities within the simulation environment.

The main data functions (Figure 42, point 1.1.) perform a variety of tasks on the data necessary for analysis as well as for preparation, conduct and report of simulation runs.

However, some initial work on the data (e.g. extraction from original data sources, initial filtering of data) has been performed using tools like standard database management systems or professional text editors.

The following functions were applied to the resulting data set.

‘Update flight DF factor (1.1.1.1.)’ works on the element of  $F_{FLIGHT}$  as introduced in Chapter 3.5. It uses flight-specific data provided by the retailing business unit and updates each record of the overall data set (summer season, winter season) with most current retail information. This function can also be used to run a ‘what-if’ analysis in which specific flights (perhaps with additional marketing support) are tested to perform differently from observed data.

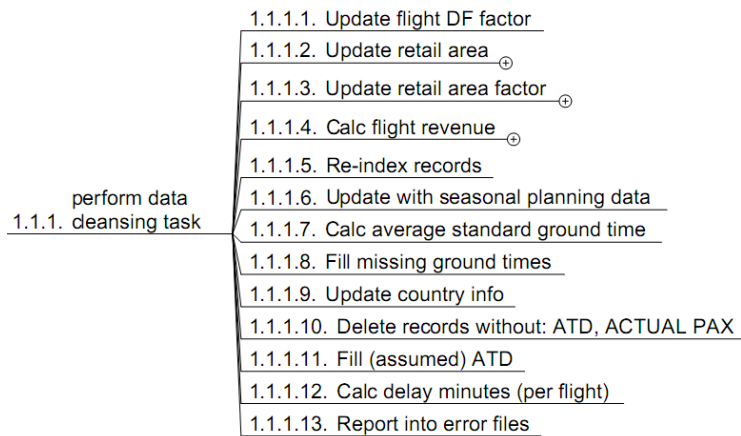


Figure 43: Functions for data cleansing.

The ‘Update retail area (1.1.1.2.)’ function determines the retail area a flight belongs to and enters this information into each record of the overall data set. There are two occasions when this function needs to be run:

<sup>94</sup> Here the term ‘function’ is used in its general meaning of functional element, comprising both functions and procedures.

- after changes in flight schedule in terms of new flights in the schedule
- after the definition of a retail area has been modified (e.g. gate 'C1' no longer belongs to retail area 'R3', but to 'R4')

After an entry of the retail area has changed (for a record) another function needs to be run, in order to maintain logical integrity. The software offers to perform the '*Update retail area (1.1.1.3.)*' function automatically. The currently defined factor ( $F_{AREA}$  as described in Chapter 3.5) is entered into the record depending on its current entry of retail area. For both functions (1.1.1.2. and 1.1.1.3.) it can be chosen, whether to update values for actual flight data or for seasonal planning data. The same applies to the '*Calc flight revenue (1.1.1.4.)*' function. It updates the entries for the revenue (sales) generated by a flight according to the data in its record. As the original set of data had to be filtered, truncated and otherwise modified several times within the data cleansing task, an updated index helped to keep track of that task. As mentioned in a previous chapter, an index is used to make data anonymous. Function '*Re-index records (1.1.1.5.)*' simply updates that index.

As it is a goal to provide simulation results for a year of data that reflects as close as possible actual flight data, the seasonal planning data had to be worked into it. Unfortunately, such planning data usually comprises only a (reference) week of flight schedule. Thus the planning data had to be mapped against the overall current data set (summer season and winter season). This is accomplished by the function '*Update with seasonal planning data (1.1.1.6.)*'.

Some flight records were missing the information of a standard ground time. But as this information is important in the gate allocation process, realistic values had to be determined for those cases missing that information. The function '*Calc average standard ground time (1.1.1.7.)*' determines such a value for each aircraft type, based on the average of standard ground times of known flights. Those values are then stored in a table for later use in the function '*Fill missing ground times (1.1.1.8.)*'.

Having introduced above that the retail potential depends on a flight's destination country, this information is crucial to the overall simulation task. Unfortunately, this information was not included in the data given. Hence, this information had to be generated via a 'linking' table that included destination (airport) information along with country information. The function '*Update country info (1.1.1.9.)*' enriches the overall data set with this information.

Flights that did not have an actual time of departure or no passengers on board, have been excluded from that data set using *'Delete records without: ATD, ACTUAL PAX (1.1.1.10.)'*<sup>95</sup> Usually, these flights had been cancelled well in advance or had been renamed as different flights. However, the data entries in the original set had not been deleted before having made available to this research.

For those flights that did not have an entry for actual time of departure (ATD), but did provide an entry for actual passengers (ACTUAL PAX), either the entry for estimated (ETD) or – as 2<sup>nd</sup> priority – scheduled (STD) time of departure has been taken as an ATD. This is accomplished by the function: *'Fill (assumed) ATD (1.1.1.11.)'*

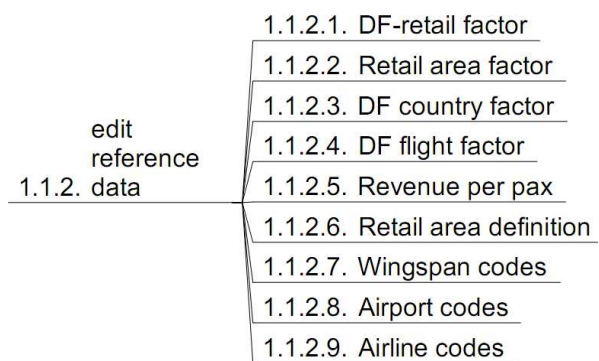
In order to test flight data and sales data for correlation regarding any delays incurred, function *'Calc delay minutes (per flight) (1.1.1.12.)'* determined delay minutes on a per flight basis.

In order to detect cases that might not have been covered by the data cleansing functions different queries have been run on the most current data set. Function *'Report into error files (1.1.1.13.)'* e.g. summarizes possible integrity problems into different files for later inspection.

As mentioned before, the data cleansing task has been very time consuming but constitutes the foundation for any further analysis.

In order to change parameters, which contribute to the quantitative research model and which reflect changes in the business environment, the simulation workbench offers a set of simple functions to perform that task.

Functions (1.1.2.1.) to (1.1.2.5.) allow for editing of factors as described along with the quantitative research model. The Function *'Retail area definition (1.1.2.6.)'* determines, which gate belongs to which retail area. After the definition of such an area has changed, all dependent functions need to be run accordingly.

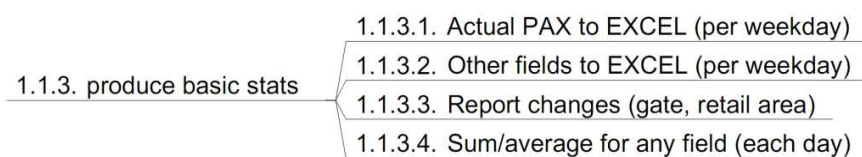


**Figure 44: Functions to manage reference data.**

<sup>95</sup> Deletions counted for less than 0.1 per cent of the data set.

As all of the sub-functions of '*edit reference data (1.1.2.)*' basically enable editing of data in text files they do not automatically cope for integrity. Therefore, it is required that the user (researcher) addresses it.

In order to produce individual data items for basic descriptive statistics they have to be extracted from the data set. Any field in question can be exported to a file that may be read by third party software. The level of aggregation and the export format is designed in a way to enable standard spreadsheet software (e.g. Microsoft Excel or Sun's OpenOffice Calc) to read the data.<sup>96</sup>



**Figure 45: Functions to produce basic statistics (descriptive data).**

Sometimes airports use the number of gate changes as an indication for planning quality. This can be misleading in case it is not very well defined, at what time and for what reason a gate allocation has been changed (incl. the complete change history of a flight). However, the retailer will only be in a position to address changes in passengers (distribution) until a certain point in time. So, the more stable an allocation plan is compared to (then) actual operations, the higher is a plan's quality from a retail perspective. Notably, the element to compare is not a gate, but a retail area. Consequently, the function '*Report changes (gate, retail area) (1.1.3.3)*' reports the degree to which gate changes lead to changes in retail area allocations.<sup>97</sup>

Usually, visualization is a good means to provide an overview of a system. Secondly, categorization of a (theoretical) system's elements helps to simplify complex structures or content. For this reason a function has been integrated to visualize various fields of flight data using colour-coded categorized information on a sort of Gantt chart.



**Figure 46: The heat map function.**

<sup>96</sup> Without any form of aggregation the total of almost 230,000 data records would not be able to be imported into e.g. Excel, because the maximum amount of approx. 65,000 lines would have been exceeded.

<sup>97</sup> For an analysis, see below Chapter 4.1.7.3.



After a selection of the field to be visualized, the values for the categories have to be defined. For example, an absolute retail revenue (sales) value per flight of 270 to 300 (Euros) may define an average. This is referred to as the ‘B’ category. Only the ‘B’ category needs to be defined. For this specific field higher values are regarded to as better results. Therefore, values of more than 300 (Euros) will define category ‘A’ whereas those of below 270 (Euros) define category ‘C’. The visual representation will be colour-coded as green (A), yellow (B) and red (C).

After having entered the start date, a week of data will be analyzed and output is generated in form of text files (in delimited format for easy spreadsheet import). Those text files contain the small letters ‘a’, ‘b’ and ‘c’ at positions where there is a flight (at intersection of time interval and gate). This form of output allows standard spreadsheet software to import and automatically colourize the data. In Appendix B, the Chapters 9.1 and 9.2 provide an example of the above.

However, the same output is also used for an internal visual representation of the categorized data. Furthermore, the seven days generated can be view quickly one after another in a sort of animation. This feature allows for easy recognition of stable portions within the data, as well as for areas of e.g. under-performance.

The heat map function is useful to obtain a brief overview of the situation.

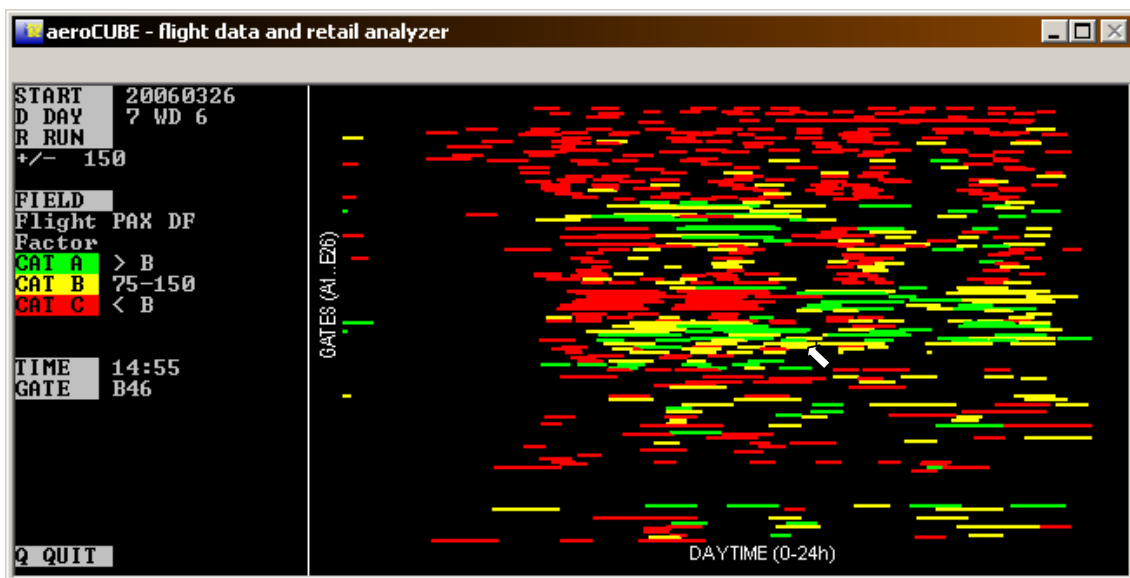


Figure 47: Screenshot of animated categorized data.

The mouse cursor can be moved around in the map. Information about its ‘logical position’ is shown on the left hand side of the screen (TIME, GATE). For example, Figure 47 shows the following situation:

The data set contains the week starting on March, 26<sup>th</sup> 2006. The current day shown is the 7<sup>th</sup> day of the data set, representing weekday 6 (a Saturday). The field is that of the flight-specific duty free factor ( $F_{FLIGHT}$ ). Category 'B' is defined as 75 to 150. The mouse cursor is located over the logical position of gate 'B46' at a time of '14:55'.

Even the simple map as shown in Figure 47 helps to identify the different traffic waves (especially to be observed at the B-gates with much international/intercontinental traffic whereas constant feeder traffic at the A-gates along with the associated time shift makes it slightly harder to determine the traffic waves in that area).

In a consulting context a different form of visualization (geographical representation) of the same data might be a more appropriate approach.<sup>98</sup>

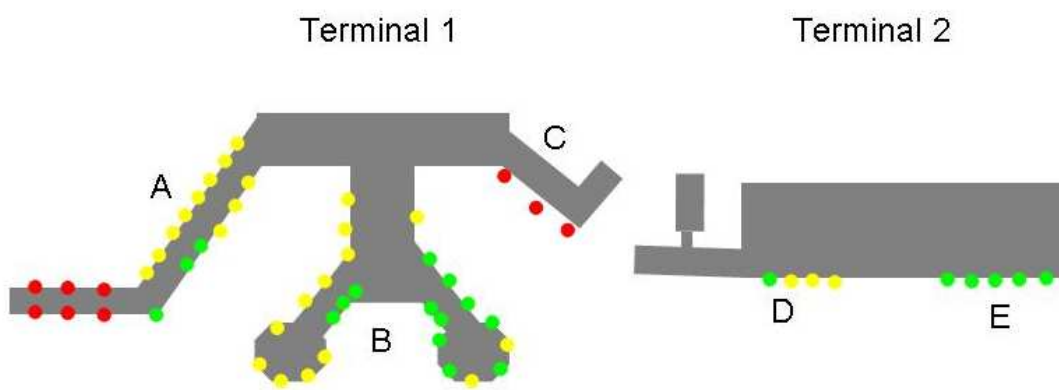


Figure 48: Geographical representation of categorized data.

Finally, the simulation workbench makes use of various support functions (Figure 42, point 2). Those functions support internal tasks and the development of the software itself. Secondly, they consist of many routines for conversions, calculations and translation of many 'real word' object names to their internal (computable) representations. An important support task has been implemented in most functions for the data cleansing task: a safety feature, which is to shift data that will be replaced by a newer version into a directory for historiography.

In order to implement the simulation workbench in a first step it had to be developed and then to be set operational on standard personal computer hardware.

<sup>98</sup> This approach is not (yet) implemented in the software.

### 3.7.2 Implementation

The Software development environment looked for had to comply with the requirements as defined at the beginning of Chapter 3.7. Additionally, from a development perspective, the following requirements had to be satisfied:

- Support of a structured general purpose programming language
- Availability of standard elements of an integrated development environment (IDE) like syntax-colouring, debugging, integrated start of compilation runs
- Production of 32 bit executable code for corresponding Microsoft Windows platforms that
  - is very fast at run-time
  - makes very efficient use of internal memory
  - allows for large amount of internal memory to be used for variables
  - allows to communicate with other Windows software by standard means like OLE (COM automation)
- Support of standard elements of a graphical user interface (GUI) like dialog boxes or menus
- Support of individual graphical elements to be defined (e.g. bars, lines)
- Avoidance of any Microsoft Windows (operating system) administrative overhead (e.g. for windowing technique)
- Ability to use standard dynamic link libraries (DLLs) in case of certain functionality would be available from a third party provider

From past experience, and after having checked against all requirements the PowerBasic development environment with the JellyFish pro editor has been chosen for the development task. The appropriate compiler within the PowerBasic family is one to generate 32 bit windows console applications.<sup>99</sup> This way it was possible to avoid speed disadvantages that usually come along with standard Windows applications, but still be able to make use of a standard graphical user interface and of the wide availability of the Windows platform. Figure 49 provides a sample screenshot of source code in the integrated development environment. It displays 60 out of approx. 10,300 lines of code. Without navigation support it would not have been possible to develop the software in just a couple of weeks. The source code listing can be found in Appendix A.

---

<sup>99</sup> The product is called PowerBasic Console Compiler 4 ('PB/CC').

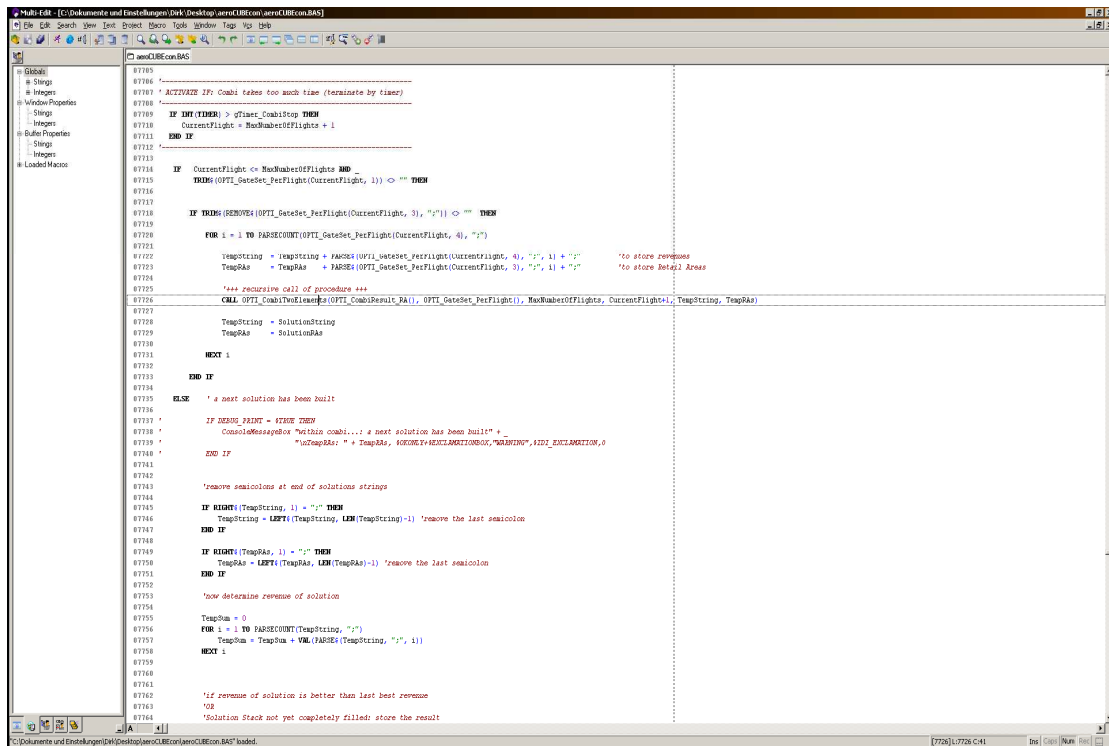


Figure 49: Example of source code displayed within integrated development environment.

The resulting executable software has the size of just about 227 kilobytes of disk space. During a simulation task it consumes approximately 5 megabytes of internal memory (RAM). This allows to run it on virtually any 32 bit Windows machine.

As a simulation run may take a long time to complete (depending on the schedule, the settings and number of days to be allocated), two aspects have found to be important and addressed within the software:

Firstly, there may be the danger of any unexpected event that could cause the software to discontinue (e.g. internal software error, power failure, hardware failure). To cope for such a case the results of a completed day of simulations are written onto hard disk. This way only the last day worked on may be lost in case of an unexpected stop of the software. A simulation run can be continued right after the last day known to be valid.

Secondly, the user is informed regarding any progress of the current simulation run. As shown in Figure 50, there are six core information elements:

- (1) Shows the *current day* that an allocation is worked on
- (2) Shows the *time* when the current simulation run *started*
- (3) Shows the *specific time interval* (1 to 288) that is worked on
- (4) Shows the number of flights that need allocation in this time interval (i.e. those not allocated in previous interval and those scheduled for departure in the current time interval)
- (5) Shows the flight numbers (internal anonymous reference number) that apply for allocation in the current time interval
- (6) Shows under each of the flights from (5) the (amount) number of eligible gates, which that flight may be allocated to

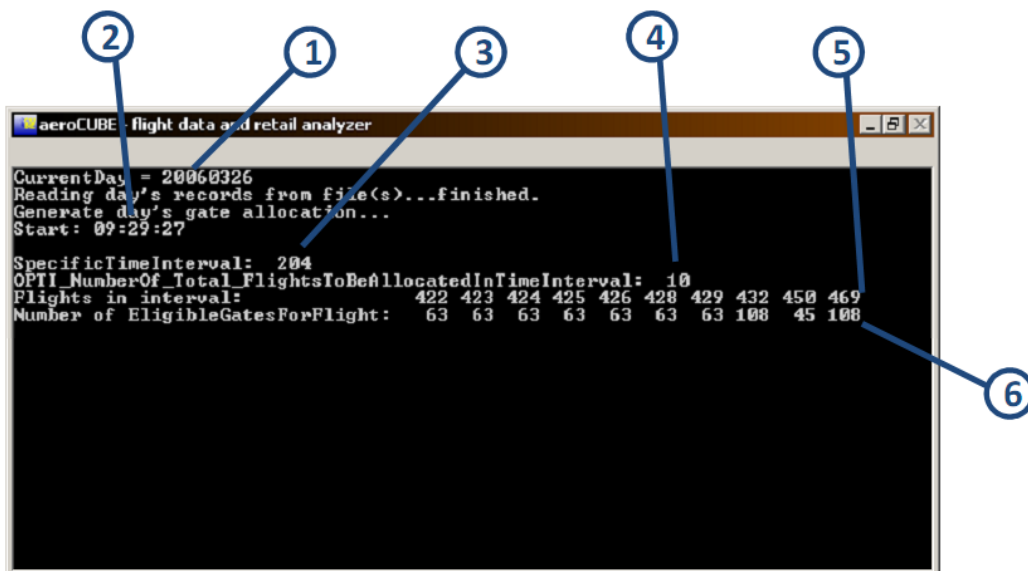


Figure 50: Sample display output during a simulation run.

For example, the situation as shown in Figure 50 provides information about the theoretical size of the solution space. Without any of the earlier introduced improvement methods the number of possible solutions would be<sup>100</sup>:

$$\begin{aligned}
 & 63 \cdot 63 \cdot 63 \cdot 63 \cdot 63 \cdot 63 \cdot 63 \cdot 108 \cdot 45 \cdot 108 \\
 = & 63^7 \cdot 108^2 \cdot 45 \\
 = & 2,067,492,157,885,974,960
 \end{aligned}$$

<sup>100</sup> Before allocation of the first flight in this interval there are 63 possible gates for each of the flights #422, #423, #424, #425, #426, #428, #429, and 108 possible gates for flight #432 as well as for flight #469, and 45 possible gates for flight #450.

However, with reduction of the solution space and run on basic personal computer hardware it usually took 3 to 10 seconds (60 as a maximum) to finish the allocation task for such an interval.

The software allows to be run independently in multiple instances on the same machine at the same time. This in consequence allows for a simulation to be either spread into separate parts, or separate (different) simulations to be run concurrently. Either way enables to make full use of multi-core processors. During simulation runs for the research project usually three of the four cores of the processor installed have been used for computing.

Summarizing Chapter 3, it can be noted that a detailed view on the business context (both in terms of processes and data) helped to formulate a conceptual research model.

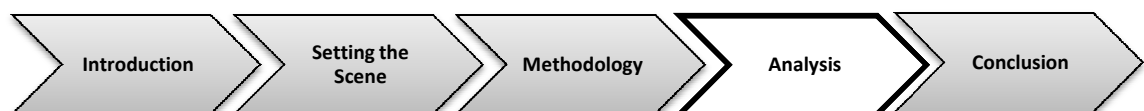
Then having identified the information needed helped to refine the conceptual model towards a quantitative research model, fulfilling objective (O.3). The type and size of the research problem required a highly improving approach that basically consists of a newly developed multi-phase gate allocation algorithm with optimistic elements and recursive search of solution space. As introduced, such an algorithm would satisfy objective (O.4). Finally, as required according to objective (O.5), a corresponding simulation environment ('workbench') has been developed and described along with its implementation.

#### *[ALIGNMENT ASPECT]*

*The previous chapters described an IT solution that is capable of producing desired output in an environment, which has as little requirements as possible regarding computing power in order to keep investment cost and operations cost low (compare, Figure 10, (4) and (5)).*

The above enables to analyze actual flight data as well as to conduct simulations runs. The latter is necessary to verify the statement (H.1) as outlined earlier in this paper. Details on these issues will be provided in the next chapters.

## *4. ANALYSIS*



## 4. ANALYSIS

This chapter builds on top of what has been developed within the methodology. Thus the simulation workbench will be used to support analysis of actual flight data and to perform simulation runs for a sensitivity analysis of retail-focussed gate allocation plans.

Finally, the objective of this analysis is an answer that aims to evaluate the statement (H.1) as introduced in Chapter 1.4.

### 4.1 Set of current flight data

As introduced in the methodology, different data sources had to be merged in order to be able to work on a single set of valid data. This data set comprises information of actual operations that were carried out during the period of observation<sup>101</sup>.

Much of that information<sup>102</sup> can be used to identify candidate parameters for the sensitivity analysis. For this reason, aspects of seasonality and possible correlations have been looked at in more detail.

#### 4.1.1 Seasonality

The business environment literally defines a certain degree of seasonality – the flight plan season. In order to cope with different demand, the transport industry has emerged to conduct business in two seasons per year. Notably, this does not reflect a calendar year but two flight plan seasons last for 12 months. A summer season usually starts in spring and ends in autumn whereas the winter season spans over the remainder (autumn to spring of the next year). Thus it needs to be considered that comparisons will only be valid within a season or when expressly different seasons are to be compared. For example the comparison of the month February to the month of July would not be valid, because they are in different flight plan seasons. However, a comparison of the two might very well be valid when the objective would be to compare a month in winter to a month in summer. In most cases comparisons will be on a seasonal correct rolling basis. In the above example, a usual comparison would be to compare February of a year to February of another year.

---

<sup>101</sup> Introduced in Chapter 3.4: 229,430 records; 195 airlines; 1,597 flights (summer); 1,102 flights (winter); 219 aircraft types.

<sup>102</sup> In addition to the results of the qualitative analysis performed within the methodology chapters – the business process decompositions.



As introduced earlier within the research project the period of time looked at spans over:

- summer season 2006 (2006-03-26 to 2006-10-28)
- winter season 2006/07 (2006-10-29 to 2007-03-24)

The next level of detail in terms of seasonality as defined by the business environment is that of a week. A flight plan for a season basically consists of multiple (same) weeks. For example, a flight ‘XY007’ may operate on days<sup>103</sup> 2, 3, 4, 5, during winter season and on all seven days of a week during summer season. Such a week is then usually applied to all weeks during the corresponding flight plan season (see Figure 51). However, actual operations are subject to interference for reasons of weather, technical issues or else.

In order to obtain an allocation planning result that is as close as possible to actual operations, for the seasonal flight plan data it has been tried to match flights with those of actual operations.<sup>104</sup>

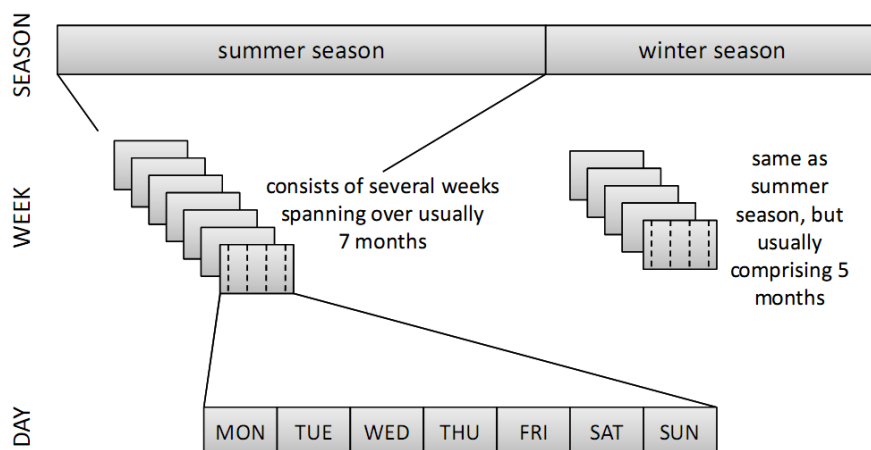


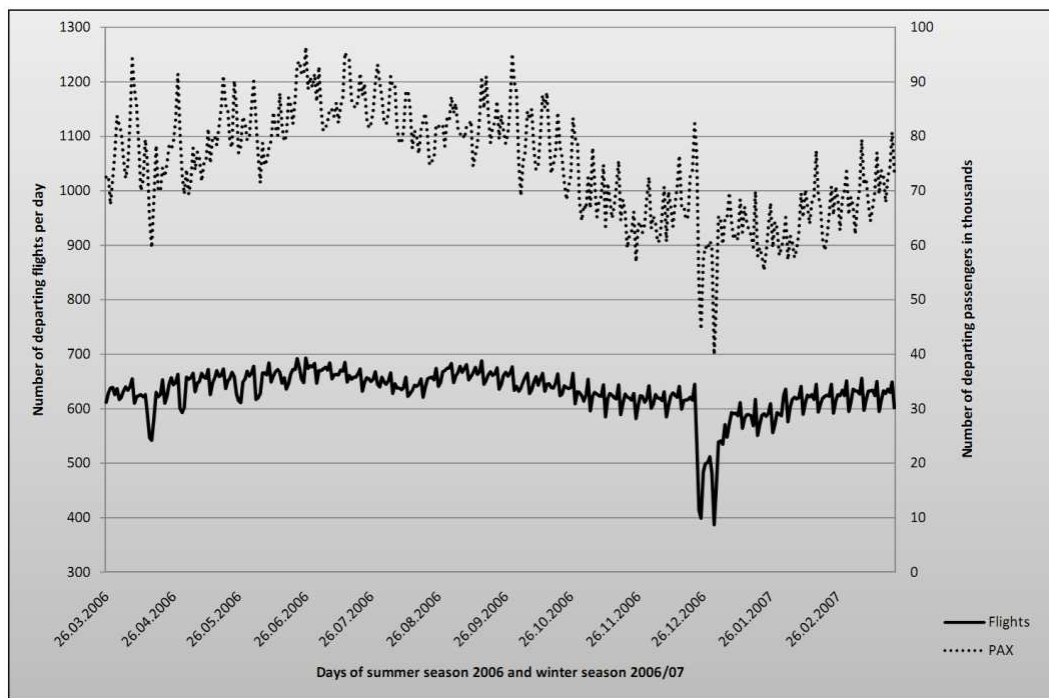
Figure 51: Formal seasonality in the research context.

The above describes formal seasonality as defined by the business environment itself. However, there are more seasonal elements to be discovered in the data provided.

<sup>103</sup> Days of a week are usually referred to as numbers, e.g. 1=Monday, 2=Tuesday, and so on.

<sup>104</sup> This has been performed within the data cleansing task.

The actual distribution of passengers and flights<sup>105</sup> is shown in Figure 52. It can clearly be differentiated between summer season and winter season. The latter incorporates lower values for both flights and passengers. An ‘outlier’ can be observed during days of Christmas at the end of the calendar year.



**Figure 52: Seasonality within actual data (summer and winter).**

As explained above, there is no seasonal interval for a month but for a week. Figure 52 shows those weekly intervals. The observed values for flights tend to be more stable than those for passengers. This is expected to be, because the variation of passengers on board of a flight is higher than that of flights themselves. On a weekly basis, different values for flights would only occur in case of cancellations or unplanned (co-ordinated on short notice) flights.

<sup>105</sup> As explained along with the research model, only departing flights are looked at. If not stated otherwise the term ‘flight’ will refer to a ‘departure’ or the departing portion of a transit or transfer flight.

A drill down into data of the above figure provides a better view on the weekly seasonality.

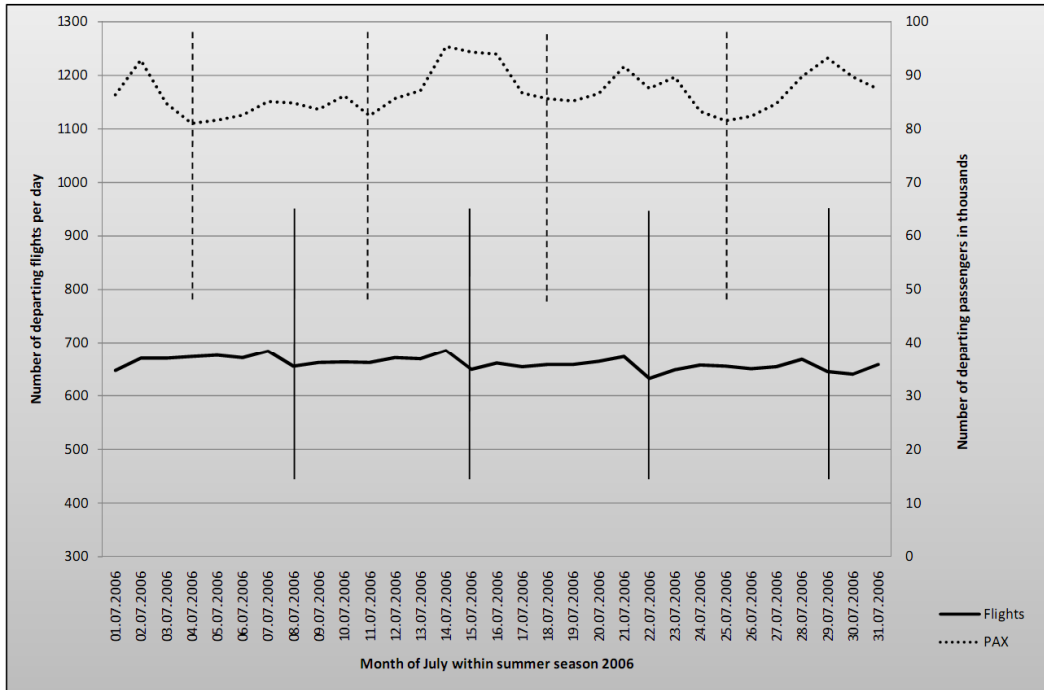


Figure 53: Seasonality within actual data (weeks in July).

The abovementioned differences in the values for flights and passengers can be clearly observed in Figure 53. If taken the maximum values for both flights and passengers the peak falls on the same day of a week (Friday). This is different should the minimum values be applied. Nevertheless, a weekly seasonal component is observed very well.

In addition to the above, actual flight data discloses another seasonal component. As mentioned in the chapter about methodology, the simulation workbench provides the possibility to visualize categorized data on a daily basis (see also Figure 47). Such a snapshot is provided in Figure 54.

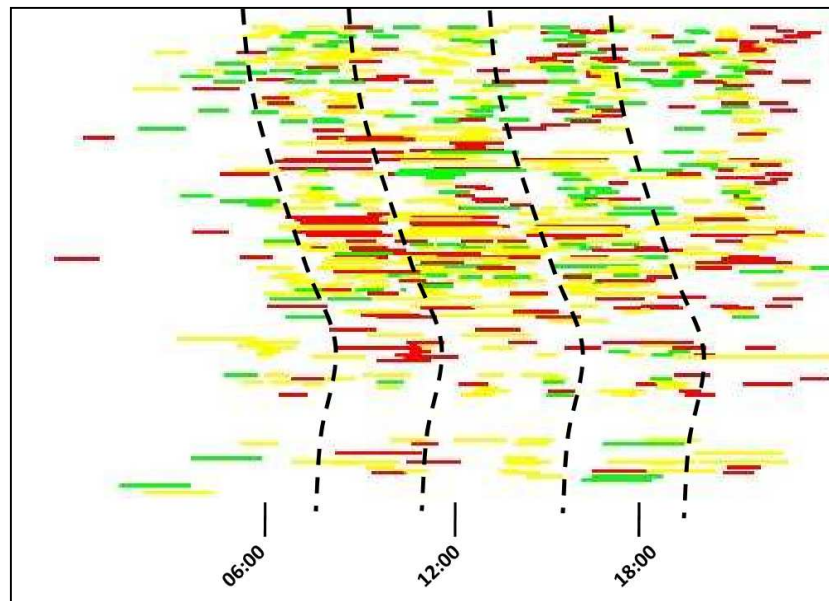


Figure 54: Seasonality within actual data (daily traffic waves).

The visualization shows data from a Wednesday (2006-07-05).<sup>106</sup>

Along the vertical direction all gates are represented, whereas horizontally the dimension of time during a day is shown. The black dashed lines indicate the boundaries between the seasonal component of a so-called ‘traffic wave’.

Interestingly, for the (allocation) improvement task, there are not necessarily those peaks within a traffic wave that require the maximum computing resources. This is due to the fact that the number of available gates is usually small during those specific time intervals. Consequently, the possible solution space is reduced.

No further seasonal components were determined. However, seasonality (as a systematic element of time series data) incorporates the dimension of time. Geographical differences regarding an object looked at are not subject to seasonality.

<sup>106</sup> Categorized data represents departure delays where colour-code yellow means a delay of 14 to 29 minutes. For the purpose to identify seasonality, a categorization is not necessary.

For example, in the abovementioned context the airport as a whole has been the object of investigation (and not a certain part of it).

Nevertheless, the major concern within the research topic is retail sales, and thus the spread of traffic over retail areas is of interest and thus provided below.

#### 4.1.2 Traffic distribution (passengers, flights)

Based on the definition of retail areas (see Table 3) the distribution of passengers provides a first view on the utilization of (gate) resources.

|          | Mon       | Tue       | Wed       | Thu       | Fri       | Sat       | Sun       | $\Sigma$   |
|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|
| R1       | 1,492,339 | 1,475,382 | 1,551,195 | 1,565,449 | 1,638,799 | 1,508,653 | 1,531,509 | 10,763,326 |
| R2       | 320,504   | 300,421   | 316,063   | 331,461   | 336,889   | 305,711   | 329,753   | 2,240,802  |
| R3       | 1,245,584 | 1,185,889 | 1,190,321 | 1,218,446 | 1,313,796 | 1,269,487 | 1,275,540 | 8,699,063  |
| R4       | 237,360   | 222,175   | 236,799   | 252,677   | 248,532   | 252,205   | 250,653   | 1,700,401  |
| R5       | 297,782   | 304,355   | 299,992   | 309,664   | 365,820   | 316,759   | 342,775   | 2,237,147  |
| R6       | 39,515    | 41,470    | 41,840    | 42,535    | 52,641    | 31,088    | 38,391    | 287,480    |
| R7       | 179,708   | 175,364   | 185,757   | 184,297   | 240,461   | 226,392   | 221,501   | 1,413,480  |
| $\Sigma$ | 3,812,792 | 3,705,056 | 3,821,967 | 3,904,529 | 4,196,938 | 3,910,295 | 3,990,122 | 27,341,699 |

Table 12: Distribution of departing passengers across retail areas for each day of a week (both seasons).

Table 12 shows the distribution of passengers (an indication for traffic) across the retail areas defined. Individual figures vary considerably. In average, fewest passengers have been observed on a Saturday in retail area R6 and most on a Friday in retail area R1. So, utilization varies much amongst the different areas.

The sum values (or presented as percentage in Figure 55) show that the vast majority (approx. 72%) of passengers use two retail areas both in Terminal 1.

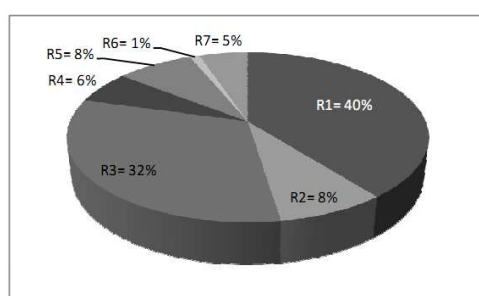


Figure 55: Distribution of passengers (sums) per retail area.

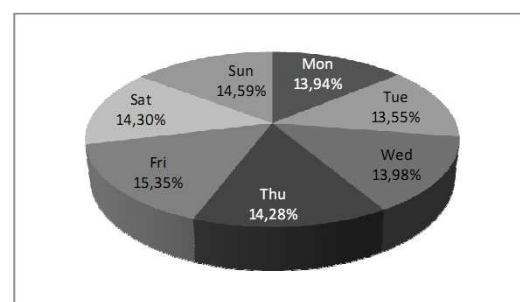


Figure 56: Distribution of passengers (sums) per day of week.

No such differences can be observed regarding the spread across the days of a week. Friday is in terms of passengers the strongest day, whereas Tuesday accounts for the weakest (Figure 56).

However, these figures represent the entire period of time looked at (summer season and winter season). Therefore, they seem to be quite evenly spread.

As introduced in Figure 52 and Figure 53, the number of flights also follows the aforementioned seasonality. Flights are spread in a similar way over a week. Based on the average number of flights per day, the strongest day is found on a Monday (693 flights) and the weakest on a Sunday (387 flights). However, the mode values explain the traffic distribution closer to reality, because they incorporate the frequencies of occurrence. Here, Friday is the strongest day (665 flights) and Saturday accounts for the weakest (617 flights). Table 13 summarizes the above.

|      | MON        | TUE | WED | THU | FRI        | SAT        | SUN        |
|------|------------|-----|-----|-----|------------|------------|------------|
| MAX  | <b>693</b> | 678 | 679 | 692 | 688        | 655        | 670        |
| MIN  | 399        | 484 | 499 | 501 | 512        | 482        | <b>387</b> |
| MODE | 633        | 638 | 634 | 626 | <b>665</b> | <b>617</b> | 648        |

Table 13: Distribution of (daily average) number of flights per weekday.

The above insight is useful input for a shortened approach towards seasonal flight planning, as the strongest day(s) may be taken as an indicator for the feasibility of a draft flight plan.

Nevertheless, it is the aim to improve the retail result by means of an adjusted gate allocation. Therefore, the individual retail areas have been looked at in terms of their relative sales performance.

### 4.1.3 Retail area factors and sales figures (model calibration)

As mentioned above, the spread of flights and their relative retail performance can be observed, which is important for later simulations. The combination of actual flight data and retail data lead to the possibility to calculate an individual factor for each retail area. These factors indicate a relative retail performance. In Table 14 below the factor of 1.00 represents as a baseline the retail area with the smallest flight-specific retail performance found.<sup>107</sup>

| Summer season |        |         |           |          |          |
|---------------|--------|---------|-----------|----------|----------|
| Retail Area   | Factor | Flights | % Flights | Weighted | % Weight |
| R1            | 1.00   | 59,194  | 41.99     | 59,194   | 30.01    |
| R2            | 1.11   | 18,415  | 13.06     | 20,441   | 10.36    |
| R3            | 2.13   | 34,921  | 24.77     | 74,382   | 37.71    |
| R4            | 1.27   | 6,908   | 4.90      | 8,773    | 4.45     |
| R5            | 1.43   | 14,292  | 10.14     | 20,438   | 10.36    |
| R6            | 1.55   | 528     | 0.37      | 818      | 0.41     |
| R7            | 1.97   | 6,704   | 4.76      | 13,207   | 6.70     |
|               | 1.40   | 140,962 | 100.00    | 197,252  | 100.00   |

| Winter season |        |         |           |            |          |
|---------------|--------|---------|-----------|------------|----------|
| Retail Area   | Factor | Flights | % Flights | Weighted   | % Weight |
| R1            | 1.00   | 36,520  | 41.28     | 36,520.00  | 28.68    |
| R2            | 1.08   | 13,081  | 14.79     | 14,127.48  | 11.10    |
| R3            | 2.24   | 22,126  | 25.01     | 49,562.24  | 38.92    |
| R4            | 1.55   | 4,306   | 4.87      | 6,674.30   | 5.24     |
| R5            | 1.31   | 6,023   | 6.81      | 7,890.13   | 6.20     |
| R6            | 1.67   | 3,771   | 4.26      | 6,297.57   | 4.95     |
| R7            | 2.37   | 2,641   | 2.99      | 6,259.17   | 4.92     |
|               | 1.44   | 88,468  | 100.00    | 127,330.89 | 100.00   |

| Both seasons |             |         |           |          |
|--------------|-------------|---------|-----------|----------|
| Retail Area  | Weighted F. | Flights | % Flights | Weighted |
| R1           | 1.00        | 95,714  | 41.72     | 95,714   |
| R2           | 1.10        | 31,496  | 13.73     | 34,568   |
| R3           | 2.17        | 57,047  | 24.86     | 123,944  |
| R4           | 1.38        | 11,214  | 4.89      | 15,447   |
| R5           | 1.39        | 20,315  | 8.85      | 28,328   |
| R6           | 1.66        | 4,299   | 1.87      | 7,116    |
| R7           | 2.08        | 9,345   | 4.07      | 19,466   |
|              |             | 229,430 | 100.00    | 324,583  |

**Table 14: Retail area factors derived from actual data.**

A basic conclusion to be drawn from the above is that the retail areas differ in performance. Although R1 counts for the most flights, it is only second (after R3) when it comes to the weighted performance<sup>108</sup>. So, R3 outperforms R1 despite the

<sup>107</sup> It has to be mentioned, that retail area performance is partially explained by the retail performance of flights allocated therein. So, there is a mutual influence of flights and retail area performance. Nevertheless, a proportion thereof is also due the retail offering. The research model assumes that the retail area factors remain unchanged for different allocations.

<sup>108</sup> The term 'weighted' means that the retail area factor is taken into account. So, for descriptive purpose the absolute and relative weight of a retail area is expressed as the 'weighted performance'. The weighted factor (3<sup>rd</sup> listing of Table 14, 'Weighted F.') simply expresses a combined value of both seasons.

considerably less traffic it serves (compared to R1). Hence theoretically, in case it would be possible to re-allocate traffic to R3 retail sales should increase.

In order to obtain an idea of the sales figures for Frankfurt Airport for further calculations the following average retail area factors have been applied<sup>109</sup>:

|         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|
| R1= 1.0 | R2= 1.1 | R3= 2.2 | R4= 1.4 | R5= 1.4 | R6= 1.7 | R7= 2.1 |
|---------|---------|---------|---------|---------|---------|---------|

Basic sales figures reflecting those of Frankfurt (as to be found in their annual reports<sup>110</sup>), have been calculated. The following has been applied to calibrate the research model:

Firstly, average sales per departing passenger have been calculated, according to the quantitative research model with the following parameters:

|                           |   |  |
|---------------------------|---|--|
| <b>P</b>                  | number of passengers on a flight  | according to actual flight data  |
| <b>S<sub>DF</sub></b>     | average duty free spending of a passenger   | 1.20 EUR (1.50 as first trial value)<br>(according to model calibration) |
| <b>F<sub>DFR</sub></b>    | a factor, which expresses the relation of duty free sales to overall retail sales (including specialty retail, and food & beverage) | 1.43<br>(i.e. 70% DF, 30% Retail, F&B) <sup>111</sup>                    |
| <b>F<sub>FLIGHT</sub></b> | a factor, which expresses the flight-specific retail behaviour of its passengers  | according to retail data   |
| <b>F<sub>AREA</sub></b>   | a factor, which expresses the location-dependent retail performance of a gate (belonging to a retail area)                          | according to simulation workbench software (R1 to R7, see above)         |

The choice of **S<sub>DF</sub>** and **F<sub>DFR</sub>** really depends on the data available for the airport in question. For example, in case there were no **F<sub>DFR</sub>** available at all, but **S<sub>DF</sub>** would be known, **F<sub>DFR</sub>** can be set to 1.00 and the model still works. Therefore, the exact values of **S<sub>DF</sub>** and **F<sub>DFR</sub>** will not influence the overall result, as long as their product express the average retail spending behaviour at that airport (not yet taking into account flight-specific and location-dependent retail performance).

<sup>109</sup> The *average* retail area factors (rounded values from Table 14) have been applied to cope for the complete year of observations, as no sales data has been made available on a seasonal basis.

<sup>110</sup> See Fraport (2006, p. 55) and Fraport (2007, pp. 32-33).

<sup>111</sup> Centre for Airport Studies (2001, p.97): Sales per passenger on a global average, normalized to Special Drawing Rights (year 2000): DF (70%), specialty retail (15%), F&B (7%), currency exchange (8%). This figure has been used, because no other valid figure had been available to the author at the time of writing the thesis. As long as the calibration of the model parameters **S<sub>DF</sub>** and **F<sub>DFR</sub>** is carried out, there is no side-effect on the overall result (see below).



Derived from:

| Year                             | 2005  | 2006     | 2007     |  |
|----------------------------------|-------|----------|----------|--|
| Pax in Mio                       | 52.2  | 52.8     | 54.2     | } According to<br>annual reports<br>Fraport AG |
| Retail                           |       |          |          |  |
| Advertisement                    | 20.60 | 27.30    | 25.50    |  |
| Services                         | 26.40 | 31.00    | 35.20    |  |
| Shopping                         | 70.90 | 76.80    | 85.10    |  |
| Sales / Pax                      | 2.26  | 2.56     | 2.69     | } Assumption<br>for airport<br>retailing       |
| Duty free (70%)                  | 49.63 | 53.76    | 59.57    |  |
| Specialty Retail, F&B (30%)      | 21.27 | 23.04    | 25.53    |  |
| Sales (Shopping) / Pax           | 1.36  | 1.45     | 1.57     | } Applied to<br>research<br>context            |
| Sales (Shopping) / Pax (Dep.)    | 2.72  | 2.91     | 3.14     |  |
| Research period                  |       | 9 months | 3 months |  |
| Shopping per Pax (Proportion)    |       | 2.18     | 0.79     |  |
| Avg. Shopping p. Dep.Pax         | ==>   |          | 2.97     |  |
| PAX (Proportion)                 |       | 39.60    | 13.55    |  |
| Total PAX (theoretical)          |       |          | 53.15    |  |
| Total Dep. PAX (theoretical)     |       |          | 26.58 *  |  |
| Total Dep. PAX (counted in data) |       |          | 27.34    |  |
| Adjust by divisor                |       |          | 1.03     |  |
| Avg. Shopping p. Dep.Pax         | ==>   |          | 2.88     |  |

\* Sum of PAX (Proportion) divided by two.

Used to cope for departing passengers only and for the time periods within 2006 and 2007.

The expected sales figure would be  $2.88 \cdot 27.34 = 78.74$  Mio Euros (78,844,664 without rounding). So, for model parameter  $F_{DFR}$  the following applies:

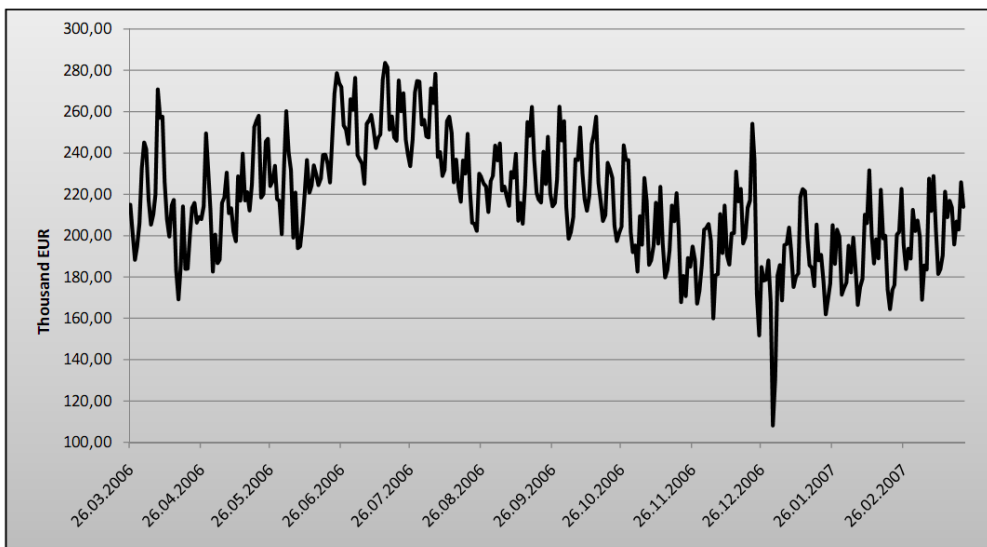
$$100\% = 2.88 \text{ (70\% = 2.02; 30\% = 0.87)}, \text{ and } F_{DFR} = \frac{2.88}{2.02} = 1.43. \text{ (see }^{112}\text{)}$$

Using  $F_{DFR}$  and a trial value for  $S_{DF}$  of 1.50 leads to an annual sales figure of 98,242,677.78 in the model. So the corrective factor for  $S_{DF}$  would be 0.80.

<sup>112</sup> In case of a different ratio (than that taken from the Centre for Airport Studies, e.g. 50%),  $F_{DFR}$  would have been different. In consequence, a different corrective factor for  $S_{DF}$  would have been calculated, leading to a new  $S_{DF}$ . However, the overall application of the two factors in the retail sales formula on a per-flight-record-basis for both seasons would lead to the same results. So, depending on the information available, and after calibration of the model, any combination of  $S_{DF}$  and  $F_{DFR}$  will lead to results as being discussed below. As long as an airport knows the average retail spending of a passenger,  $S_{DF}$  may be set to that value and  $F_{DFR}$  may then be set to 1.00.

Using the corrected value for  $S_{DF}$  ( $= 1.20$ ) results in a sales figure of 78,594,143.54, which is close to the expected figure of 78,844,664.

The above ( $S_{DF} = 1.20$ ;  $F_{DFR} = 1.43$ ) leads to a distribution of daily sales figures as to be observed in Figure 57.



**Figure 57: Retail sales for each day within research period.**

Total sales within summer season summed up to approx. 50.0 Mio Euros, and the corresponding figure for the winter season was calculated to be 28.6 Mio Euros.

More insight regarding contribution to this retail result can be gained from the next four figures showing the different countries' sales performance.

Having grouped the flights by destination countries and accumulated their sales contribution, states that (for both summer season and winter season) flights to 10 countries account for approx. 50% of sales, and approx. 80% of sales results from flights to 30 different countries of destination (see Figure 58 and Figure 59).

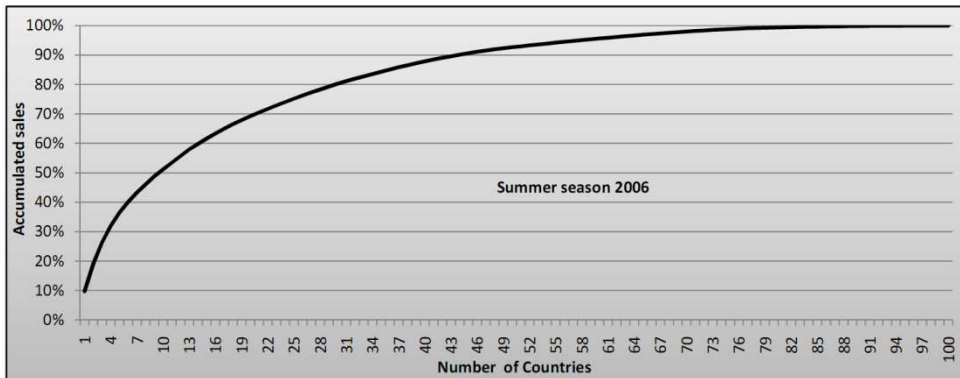


Figure 58: Accumulated sales per country (ABC-curve, summer season).

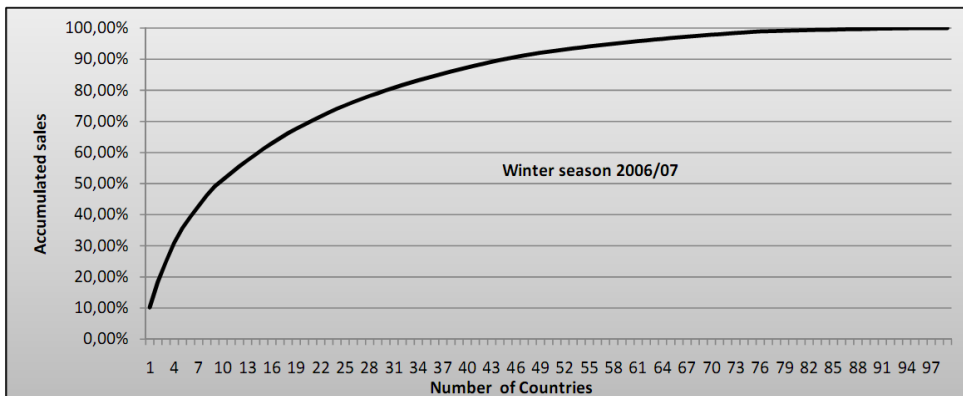


Figure 59: Accumulated sales per country (ABC-curve, winter season).

Approx. 61% of sales are produced by flights into 15 different countries. However, the percentages do not vary between summer season and winter season, but the contributors to that result are slightly different.

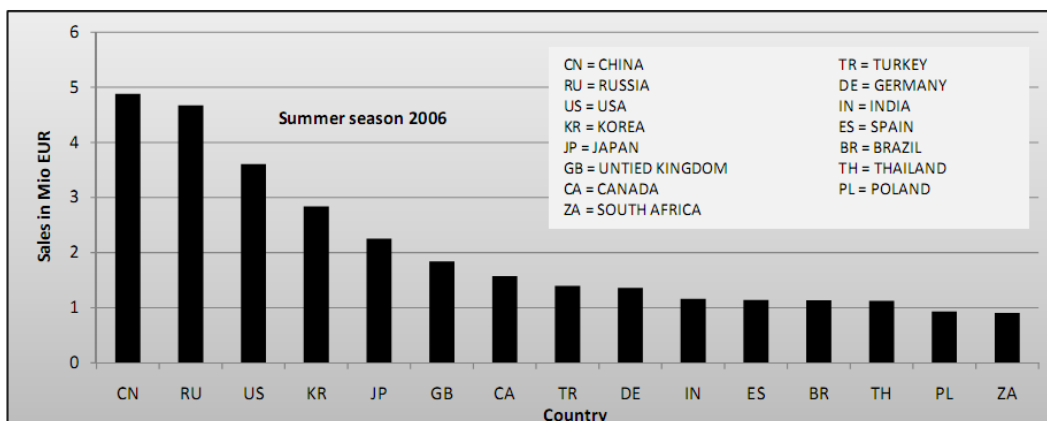


Figure 60: Top 15 countries in sales (summer season).

Whereas the most contributing positions remain unchanged, in the second half the countries' individual retail results are similar, only minor changes occur.

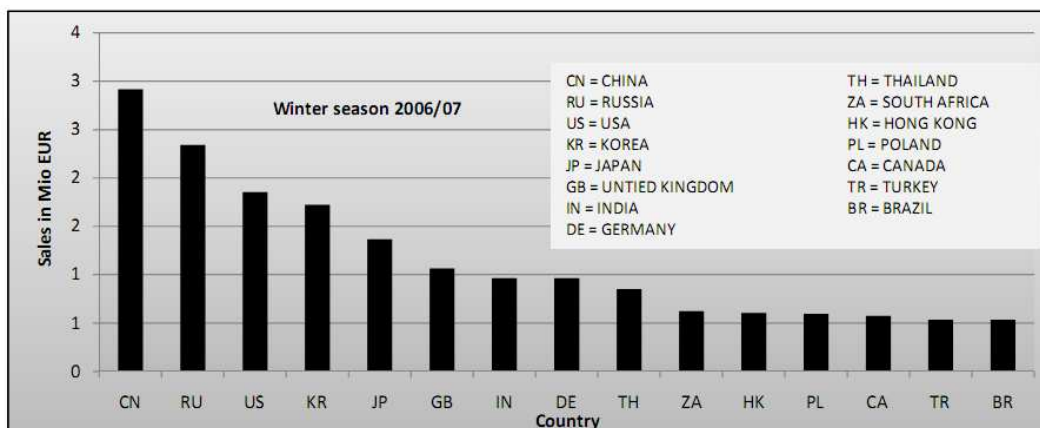


Figure 61: Top 15 countries in sales (winter season).

For example, Spain is a summer destination not found amongst the top 15 countries in the winter season. Whereas Hong Kong is not amongst the top 15 in summer, but in winter.

Taking into account that according to Table 4 some of the above countries (e.g. Spain or Germany) have poor retail-worthiness, the amount of traffic compensates for this to a large extent.

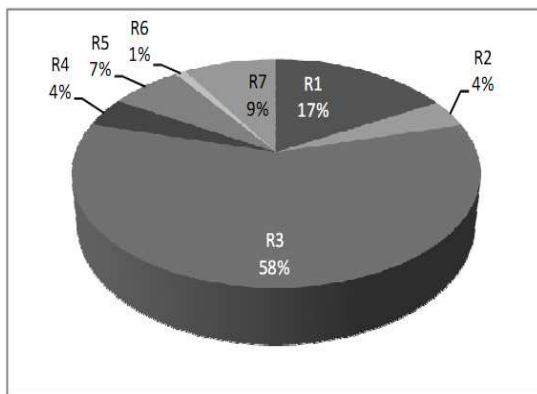
Further insight is derived from the distribution of retail sales across the different retail areas. As expected from the aforementioned (passenger distribution, different performance of retail areas) there should be considerably different figures in sales for each of the retail areas.

|    | Mon        | Tue        | Wed        | Thu        | Fri        | Sat        | Sun        | Σ          |
|----|------------|------------|------------|------------|------------|------------|------------|------------|
| R1 | 1,860,468  | 1,831,911  | 1,852,479  | 1,830,156  | 1,981,944  | 1,945,155  | 1,989,523  | 13,291,636 |
| R2 | 451,211    | 411,876    | 434,167    | 455,317    | 461,105    | 417,916    | 475,141    | 3,106,733  |
| R3 | 6,444,882  | 6,230,744  | 6,217,226  | 6,288,838  | 7,031,121  | 6,748,697  | 6,917,193  | 45,878,702 |
| R4 | 485,306    | 465,803    | 454,991    | 510,096    | 469,055    | 449,314    | 496,093    | 3,330,657  |
| R5 | 681,326    | 640,246    | 680,069    | 717,828    | 919,308    | 814,734    | 853,663    | 5,307,175  |
| R6 | 98,754     | 107,494    | 111,222    | 118,494    | 130,319    | 88,681     | 121,979    | 776,943    |
| R7 | 929,728    | 877,556    | 901,007    | 817,040    | 1,128,545  | 1,126,778  | 1,122,699  | 6,903,351  |
| Σ  | 10,951,675 | 10,565,630 | 10,651,160 | 10,737,769 | 12,121,396 | 11,591,275 | 11,976,291 | 78,595,197 |

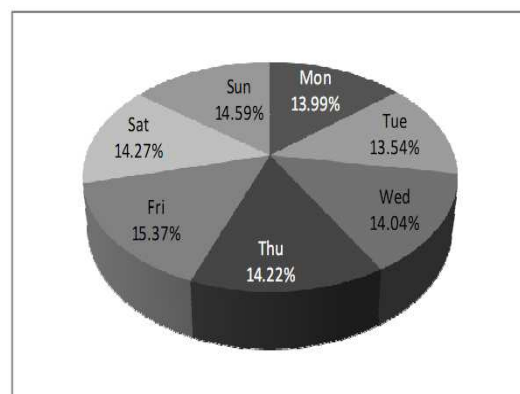
Table 15: Distribution of retail sales (in EUR) across retail areas for each day of a week (both seasons).

The minimum sales figure is observed on a Saturday in retail area R6, and the maximum is found on a Friday in retail area R3. However, the maximum number of passengers had been observed for the same day of week in R1 (compare Table 12). The performance factor of R3 leads to this better result.

Regarding their relative contribution to the overall retail result R1 and R3 are most important (see Figure 62). R1's favourable position is due to its volume of passengers. Therefore, there are basically two ways to improve the retail result. Firstly, given the retail factor of R1 is the lowest, an increase here would improve sales considerably. Secondly, traffic from R1 would need to be re-allocated towards gates in retail areas with higher  $F_{AREA}$ . A simulation of the different scenarios (introduced beginning of Chapter 4.2) will need to cope for this.



**Figure 62: Distribution of retail sales (sums) per retail area (basis: actual data).**



**Figure 63: Distribution of retail sales (sums) per day of week (basis: actual data).**

Far more homogeneous is the distribution of retail sales over the days of a week and follows very closely the corresponding distribution of passengers (compare Figure 56).

As the absolute numbers of passengers and the flight schedule as such will not be altered by any scenario the simulation results' percentage figures on a per-weekday-basis for sales are not expected to be much different from actual figures.<sup>113</sup>

The introduction of the conceptual research model mentioned how smoothly running operations are a precondition to the (improved) generation of retail sales. So, later on some selected operational figures will provide more insight into this area of research.

<sup>113</sup> Passenger figures have to be the same of course.

#### 4.1.4 Sales per passenger

Having gained knowledge about the individual absolute figures of passengers, flights and sales, there are two ratios that describe the situation regarding sales more precisely: ‘sales per passenger’ (Figure 64, Table 16) and ‘sales per flight’ (Figure 65).

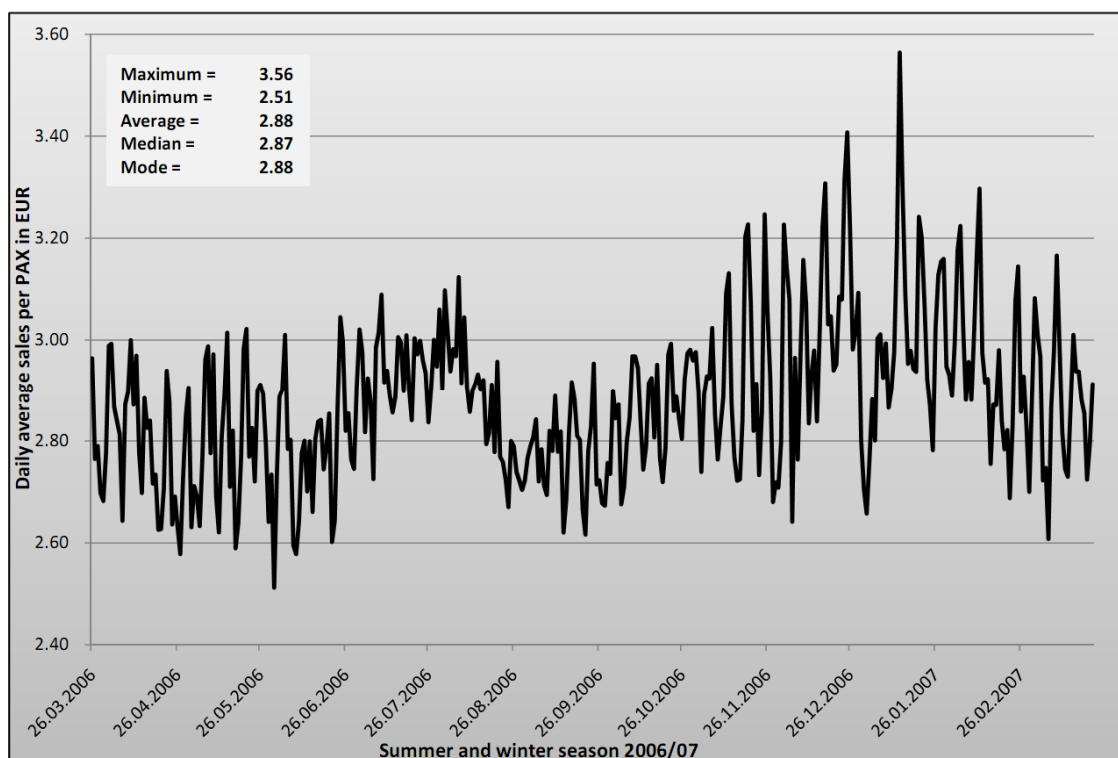


Figure 64: Sales per departing PAX, based on daily average.

The curve of average sales per passenger shows two to four inclines (two major ones: June to August, October to December). As expected, the average sales figure is about 2.88 EUR per passenger.

From a retail perspective the Saturday is close to a typical business day. Average sales is 2.88 EUR per passenger, which is the most common daily average for that figure.

Table 16 emphasizes the strong sales performance of R3 compared to R1. Despite its high retail performance (4.89 EUR per passenger), R7 only counts for 9% of overall retail sales (see Figure 62). This is basically due to the low proportion (5%) of overall passengers that are handled through gates in R7 (see Figure 55).

|      | Mon  | Tue  | Wed  | Thu  | Fri  | Sat  | Sun  | Avg. |
|------|------|------|------|------|------|------|------|------|
| R1   | 1.25 | 1.24 | 1.19 | 1.17 | 1.21 | 1.29 | 1.30 | 1.24 |
| R2   | 1.41 | 1.37 | 1.37 | 1.37 | 1.37 | 1.37 | 1.44 | 1.39 |
| R3   | 5.17 | 5.25 | 5.22 | 5.16 | 5.35 | 5.32 | 5.42 | 5.27 |
| R4   | 2.04 | 2.10 | 1.92 | 2.02 | 1.89 | 1.78 | 1.98 | 1.96 |
| R5   | 2.29 | 2.10 | 2.27 | 2.32 | 2.51 | 2.57 | 2.49 | 2.36 |
| R6   | 2.50 | 2.59 | 2.66 | 2.79 | 2.48 | 2.85 | 3.18 | 2.72 |
| R7   | 5.17 | 5.00 | 4.85 | 4.43 | 4.69 | 4.98 | 5.07 | 4.89 |
| Avg. | 2.83 | 2.81 | 2.78 | 2.75 | 2.79 | 2.88 | 2.98 |      |

Table 16: Sales (in EUR) per departing passenger per retail area and day of week (both seasons).

Consequently, the improvement algorithm is expected to exploit this potential through a retail-favoured gate allocation.

#### 4.1.5 Sales per flight

However, the smallest allocation unit is not a passenger, but a flight. In general, it would be expected that the ratio of sales per flight is similar to that of sales per passenger.

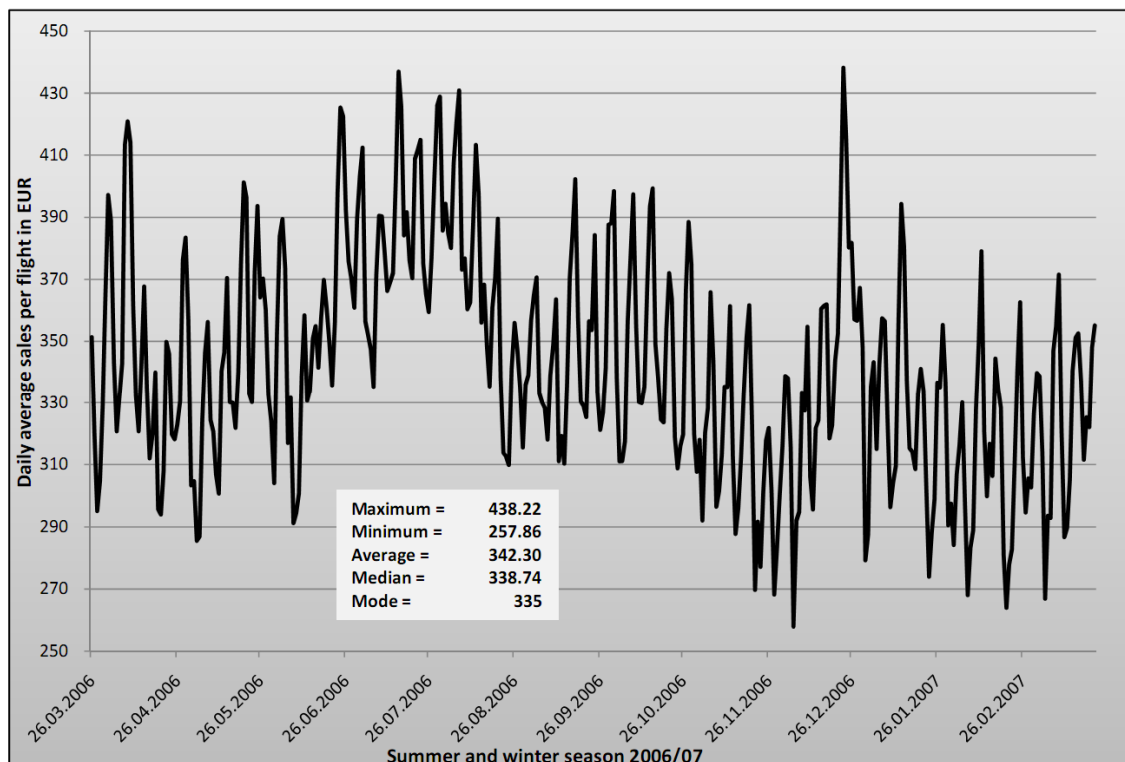


Figure 65: Sales per departing flight, based on daily average.

Nevertheless, as Figure 65 indicates in the period of September to the beginning of December a reciprocal slope (compared to passengers) can be observed. Sales per passenger increased during that time whereas sales per flight decreased. This means that in this period passenger numbers decreased to a larger extent than the number of flights decreased. The fact that the values in Figure 65 represent already daily averaged sales figures, hides extreme values for individual flights. These can be located at more than 8,000 Euros per flight.

#### 4.1.6 Comparison of retail sales (actual traffic vs. seasonal planning)

According to the conceptual research model, a retail-focussed gate allocation ought to be the basis for seasonal flight planning. But already in the present situation there is a seasonal flight planning process in place, which produces an allocation plan. The resulting distribution of passengers over the retail areas is shown in Table 17 and Figure 66, whereas Figure 67 shows it per day of week.

|          | Mon       | Tue       | Wed       | Thu       | Fri       | Sat       | Sun       | $\Sigma$   |
|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|
| R1       | 1,428,012 | 1,415,114 | 1,451,967 | 1,564,174 | 1,545,893 | 1,381,460 | 1,440,286 | 10,226,906 |
| R2       | 354,664   | 348,018   | 376,834   | 300,852   | 374,489   | 391,043   | 401,571   | 2,547,471  |
| R3       | 1,184,595 | 1,125,444 | 1,131,091 | 1,141,209 | 1,262,165 | 1,215,761 | 1,190,492 | 8,250,757  |
| R4       | 228,011   | 201,471   | 231,705   | 231,575   | 244,033   | 229,843   | 238,587   | 1,605,225  |
| R5       | 247,315   | 238,391   | 251,513   | 270,398   | 305,900   | 246,536   | 261,448   | 1,821,501  |
| R6       | 38,083    | 40,188    | 40,962    | 43,370    | 50,122    | 31,208    | 30,703    | 274,636    |
| R7       | 178,640   | 173,109   | 190,305   | 168,153   | 238,292   | 236,784   | 254,178   | 1,439,461  |
| $\Sigma$ | 3,659,320 | 3,541,735 | 3,674,377 | 3,719,731 | 4,020,894 | 3,732,635 | 3,817,265 | 26,165,957 |

Table 17: Distribution of departing passengers (seasonal plan) across retail areas for each day of a week (both seasons).

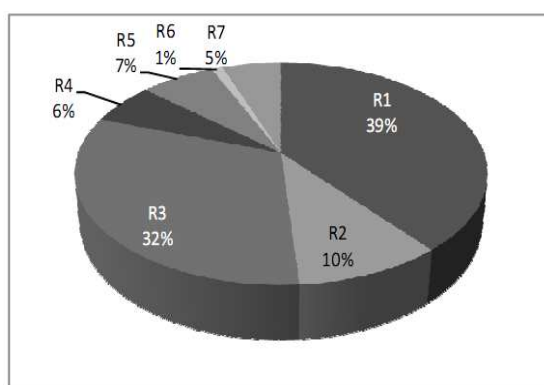


Figure 66: Distribution of passengers (sums) per retail area (seasonal plan).

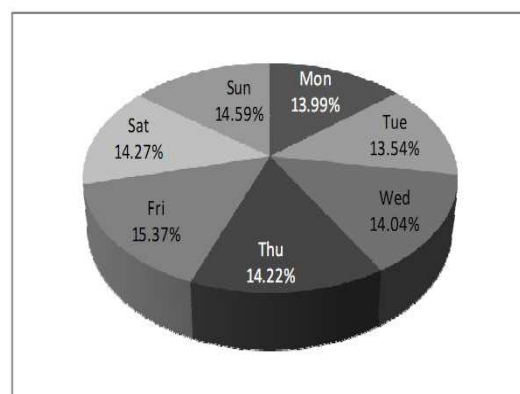


Figure 67: Distribution of passengers (sums) per day of week (seasonal plan).



Compared to the distribution of passengers from actual operations (Figure 55, Figure 56, Table 12) there is almost no difference. The slight difference is partly due to the fact that it had to be coped for incomplete seasonal flight plan data.

Only those records with entries for both actual sales and seasonal plan sales have been incorporated into the above table and figures (on seasonal plan data). Taking this as the common overlap in data, it leads to an overall retail sales result of 74,831,642 (seasonal planning) versus 75,247,166 (actual operations).

Given the more than 99% identical result may be a strong indication for the high degree of influence that the seasonal flight planning result has towards the actual sales result.

The operational explanation for this effect is provided below in Chapter 4.1.7.3.

In order to gain more insight into the performance of the operations function, selected observations are provided below.

#### **4.1.7 Figures regarding flight operations**

In the context of the conceptual research model (see Figure 31) it is spoken of 'additional cost at operations'. However, also without a focus on retailing (in the gate allocation process) there are elements that cause a burden to flight operations. Some of them have been looked at more closely, in order to determine a possible relevance for the formulation of scenarios.

### 4.1.7.1 Delay

In order to cope for a certain factor of uncertainty in operations, actual delay minutes are used as an indication.

Using the actual flight data provided, the data fields of ‘scheduled time of departure’ and ‘actual time of departure’ provide the possibility to calculate such a figure for delay.<sup>114</sup>

Divided into 11 delay classes of different duration (in minutes), Table 18 shows the distribution of frequencies of delay minutes:

| <u>Class</u> | <u>Total</u>       |
|--------------|--------------------|
|              | <u>occurrences</u> |
| 0 - 5        | 10074              |
| 6 - 10       | 24954              |
| 11 - 15      | 43012 (Mode= 15)   |
| 16 - 20      | 41936 (Median= 20) |
| 21 - 25      | 31305              |
| 26 - 30      | 21236 (Mean 26,59) |
| 31 - 60      | 43204              |
| 61 - 120     | 11099              |
| 121 - 240    | 2031               |
| 241 - 600    | 444                |
| 601 - 2000   | 135 (Max. 1801)    |
| <u>Total</u> | <u>229430</u>      |

**Table 18: Frequency of occurrences of delay minutes (both seasons).**

The information from this table is used as an input for later scenario definition. A scenario parameter representing a ‘buffer time’ will indicate that a gate needs to be blocked for a certain time after the previous flight on that gate has departed.

Thus, in order to construct a ‘close-to-real-world’ scenario and according to Table 18, there will need to be applied buffer times of 15 and 20 minutes.

<sup>114</sup> It is not aimed to match one of the several (more or less official) definitions for ‘delay’, but to provide an indication of ‘disturbance’ in operations. For the purpose of research, delays that lead to a re-scheduled flight on another day have been omitted.

#### 4.1.7.2 Turnaround times

The time an aircraft takes between on-block and off-block is referred to as turnaround time. In the flight data this is called standard ground time. The simulation will use this as follows: An aircraft arrives at the gate at a time  $t_1$  and leaves it at time  $t_2$ . The time  $t_2 - t_1$  is the standard ground time. It is assumed that an aircraft will remain at a gate during that time (not being towed away in between). The gate allocation will calculate backwards from  $t_2$ . That means the standard ground time will have an influence on the gate allocation process. In order to obtain some insight to determine meaningful values for that parameter in scenario construction, frequencies of occurrence have been computed.

|       |     | Summer Season |       |       |       |       |       |       |
|-------|-----|---------------|-------|-------|-------|-------|-------|-------|
| Class |     | MON           | TUE   | WED   | THU   | FRI   | SAT   | SUN   |
| 0     | 30  | 2318          | 2343  | 2282  | 2335  | 2347  | 1888  | 1944  |
| 31    | 60  | 13072         | 13060 | 13145 | 13023 | 13298 | 12507 | 13042 |
| 61    | 90  | 1829          | 1859  | 1815  | 1820  | 1939  | 1842  | 1720  |
| 91    | 120 | 892           | 943   | 956   | 985   | 958   | 985   | 888   |
| 121   | 150 | 965           | 1075  | 982   | 1064  | 1006  | 1028  | 1039  |
| 151   | 180 | 940           | 886   | 956   | 889   | 990   | 954   | 947   |
| 181   | 210 | 43            | 67    | 51    | 62    | 78    | 66    | 97    |
| 211   | 420 | 5             | 6     | 4     | 4     | 2     | 4     | 7     |
| 421   | 450 | 1             | 3     | 1     | 1     | 1     | 13    | 16    |
| 451   | 720 | 2             | 3     | 2     | 8     | 1     | 10    | 1     |
|       |     | 20067         | 20245 | 20194 | 20191 | 20620 | 19297 | 19701 |
|       |     | 140315        |       |       |       |       |       |       |

note: only records with an entry for standard ground time considered

**Table 19: Frequency of occurrences of standard ground time entries (summer season).**

|       |     | Winter Season |       |       |       |       |       |       |
|-------|-----|---------------|-------|-------|-------|-------|-------|-------|
| Class |     | MON           | TUE   | WED   | THU   | FRI   | SAT   | SUN   |
| 0     | 30  | 1536          | 1560  | 1572  | 1588  | 1561  | 1231  | 1227  |
| 31    | 60  | 8060          | 8188  | 8048  | 8159  | 8387  | 7728  | 8050  |
| 61    | 90  | 1087          | 1070  | 1223  | 1026  | 1259  | 1211  | 1190  |
| 91    | 120 | 546           | 625   | 579   | 593   | 660   | 532   | 477   |
| 121   | 150 | 628           | 629   | 630   | 656   | 624   | 661   | 634   |
| 151   | 180 | 566           | 509   | 574   | 516   | 571   | 528   | 548   |
| 181   | 210 | 51            | 50    | 39    | 40    | 53    | 58    | 58    |
| 211   | 420 | 0             | 2     | 5     | 4     | 1     | 0     | 6     |
| 421   | 450 | 0             | 1     | 1     | 0     | 0     | 6     | 2     |
| 451   | 720 | 0             | 0     | 1     | 1     | 1     | 1     | 1     |
|       |     | 12474         | 12634 | 12672 | 12583 | 13117 | 11956 | 12193 |
|       |     | 87629         |       |       |       |       |       |       |

note: only records with an entry for standard ground time considered

**Table 20: Frequency of occurrences of standard ground time entries (winter season).**

A number of 10 classes have been defined. Classes 1 to 9 are of 30 minutes duration, whereas the 10<sup>th</sup> class copes for all remaining flights (with upper boundary of 720

minutes). Both seasons (Table 19 and Table 20) are similar in terms of their frequency distribution across the classes.

Due to incomplete (actual) data there is a difference of 1,486 flights between total amount of flights and those with an entry for the standard ground time. Nevertheless, the data obtained is still very sufficient to get insight regarding the entry of standard ground times.

The overall frequency distribution can be observed in Table 21.

| <u>Class</u> |     | <u>Total</u>       |                        |
|--------------|-----|--------------------|------------------------|
|              |     | <u>occurrences</u> |                        |
| 0            | 30  | 25732              |                        |
| 31           | 60  | 147767             |                        |
| 61           | 90  | 20890              |                        |
| 91           | 120 | 10619              |                        |
| 121          | 150 | 11621              |                        |
| 151          | 180 | 10374              |                        |
| 181          | 210 | 813                |                        |
| 211          | 420 | 50                 | note:                  |
| 421          | 450 | 46                 | only records with      |
| 451          | 720 | 32                 | an entry for standard  |
|              |     |                    | ground time considered |

**Table 21: Frequency of occurrences of standard ground time entries (both seasons).**

For scenario construction this can be interpreted and used as follows:

If a scenario would prescribe a maximum value for standard ground time, all values higher than such a maximum value would be reduced to that value.<sup>115</sup> So, high values for standard ground time would be closer to actual traffic, whereas a value of e.g. 120 minutes would cause 22,936 flights (i.e. approx. 10%) to decrease standard ground time. But it is exactly such a reduction that generates higher flexibility in the gate allocation process, because a gate would be become vacant earlier.

A minimum value for a standard ground time means that all values below such a minimum value would be increased to that value.<sup>116</sup> So, low values for standard ground time would be closer to actual traffic (as in actual flight plan data), but they might include unrealistic short turnaround times. A value of e.g. 61 minutes would cause 173,499 flights (i.e. approx. 76%) to increase standard ground time. This of course limits the gate allocation process' ability to find retail-favoured gates, because a gate would be occupied for a longer time.

<sup>115</sup> Thus, urging an airline to increase speed in turnaround of that flight.

<sup>116</sup> This might be useful to cope for unrealistic low values in actual flight (plan) data.

#### 4.1.7.3 Gate changes versus changes in retail areas

According to the conceptual research model another aspect that needs to be looked at, is that of joint planning (between retail and operations)<sup>117</sup>. It is assumed that there is a remarkable impact of planning towards the retail result. In reverse conclusion this would assume that actual operations do not influence the planning basis to such a large extent, that a planning phase would become needless.

Therefore, the seasonal flight plan data has been compared to actual flight data. A proportion of 94% of the data contained information on both, actual gates and gates according to seasonal flight planning.

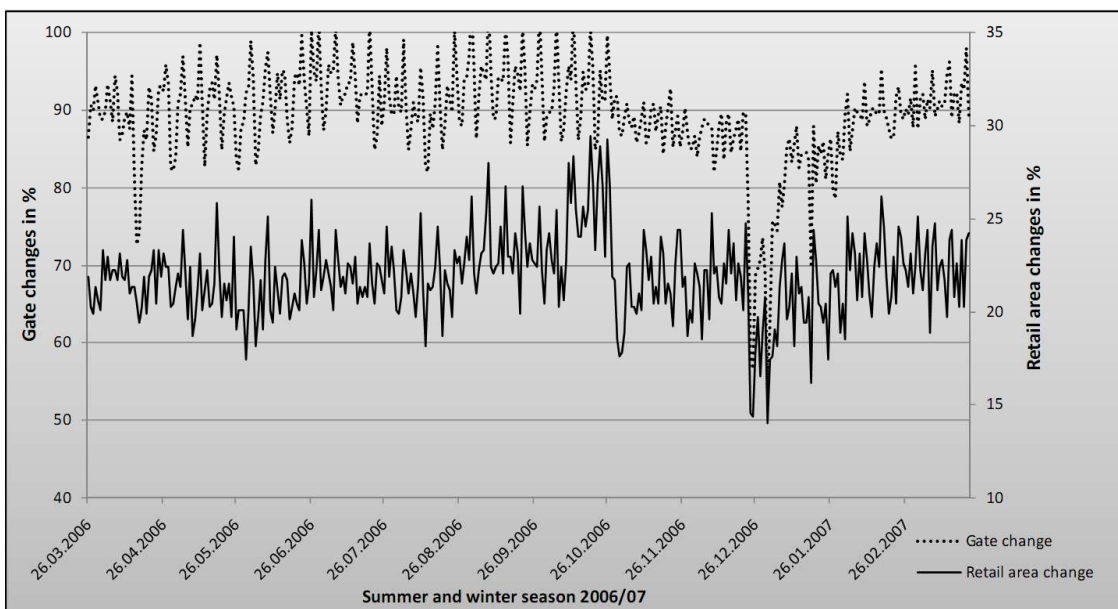


Figure 68: Gate changes resulting in changes of retail area.

Thereof 89% provided different entries for a gate (i.e. a gate change)<sup>118</sup>, but only 22% also led to a change in retail area. There is a ratio of 3.2 to 5.3 (average of 4.1) between gate change and a change of retail area.

This supports the conclusion from Chapter 4.1.6 that despite of a considerable amount of gate changes (but relatively fewer changes in retail areas) the retail result is very much pre-determined by the output of the planning process. In addition, this also backs part of the basic statement as expressed in the conceptual research model.

<sup>117</sup> Compare again Figure 30.

<sup>118</sup> At Taoyuan International Airport, Taipei (Taiwan) there is a goal value of only 10% changes between the pre-planned day and the current day of operations.

#### 4.1.8 Causality: Falsification using correlations

Finally, in order to obtain an even more complete picture regarding possible contributions towards the retail sales result, simple correlations have been determined. The purpose is not to determine any specific causality. Furthermore, it is aimed to identify, whether a variable may not be of any contribution towards the sales result. In case there was no correlation between two variables, a necessary (but not sufficient) precondition for causality would be missing. Therefore, by means of falsification, non-contributing variables may be detected.

An initial set of correlations have been based on aggregated (average per day) figures. They included figures for number of flights (FLIGHTS), number of passengers (PAX), amount of delay in minutes (DELAY), and retail sales result (SALES).

|         | FLIGHTS | PAX  | DELAY | SALES |
|---------|---------|------|-------|-------|
| FLIGHTS | 1.00    | 0.74 | 0.35  | 0.60  |
| PAX     |         | 1.00 | 0.41  | 0.92  |
| DELAY   |         |      | 1.00  | 0.39  |
| SALES   |         |      |       | 1.00  |

**Table 22: Correlation coefficients for daily aggregated (averaged) values of flights, passengers, delay (minutes), and sales.**

As can be observed in Table 22 (and as expected), correlation exists to a certain extent (not further tested for significance) between each pair of variables. As mentioned above, this does not explain any causality, but implies that the causal modal may be valid (otherwise a null correlation would have falsified the model).

A note regarding DELAY: An assumption here might be that passengers purchase more goods when their flight is delayed. However, this would require the possibility to have a shopping facility close to the corresponding gate (in case the passengers have already proceeded to the gate and wait there for boarding). So, a decentralized (gate hold room – based) retail offer might support increased sales in case of delays. Nevertheless, in many cases the pure retail shopping will already have been done until the point a passenger starts waiting in the gate hold room. So, an increase in overall spending might be in the food and beverage category. Furthermore, a causal relationship between DELAY and SALES would assume that the delay incurred prior to boarding of the aircraft. Realistically, very often boarding is on time, but while being on position or on taxiway delay minutes sum up. As the data for this piece of research provides no insight regarding the reasons for delay, the factor DELAY is not considered further in the gate allocation process.

The data basis of Table 22 is a daily average for each variable. However, the model as specified in Chapter 3.5 builds on the individual flight event. Therefore, a second ‘falsification’ attempt has been conducted. This time DELAY has not been looked at for the reasons discussed above, but the flight-specific retail factor ( $F_{FLIGHT}$ ) has been incorporated instead. The sample size consists of 229,430 events. In addition to Table 22, potential correlation has been looked at on a per-day-of-week-basis (see Table 23). Again, the causal model has not been falsified, because correlations have been determined for each pair looked at.

| SUMMER SEASON          |       | MON    | TUE    | WED    | THU    | FRI    | SAT    | SUN    |
|------------------------|-------|--------|--------|--------|--------|--------|--------|--------|
| Correlation between    |       |        |        |        |        |        |        |        |
| PAX                    | SALES | 0.5287 | 0.5254 | 0.5159 | 0.5078 | 0.5252 | 0.5235 | 0.5143 |
| $F_{FLIGHT}$           | SALES | 0.7350 | 0.7170 | 0.7193 | 0.6950 | 0.7317 | 0.6994 | 0.7418 |
| Number of observations |       | 20,139 | 20,355 | 20,268 | 20,300 | 20,691 | 19,405 | 19,804 |
| WINTER SEASON          |       | MON    | TUE    | WED    | THU    | FRI    | SAT    | SUN    |
| Correlation between    |       |        |        |        |        |        |        |        |
| PAX                    | SALES | 0.5678 | 0.5758 | 0.5636 | 0.5712 | 0.5647 | 0.5825 | 0.5783 |
| $F_{FLIGHT}$           | SALES | 0.7176 | 0.6795 | 0.6915 | 0.6430 | 0.7032 | 0.6853 | 0.7333 |
| Number of observations |       | 12,585 | 12,761 | 12,780 | 12,711 | 13,232 | 12,085 | 12,314 |

Table 23: Correlation coefficients for passengers, sales and flight-specific retail factor, based on single flight events, grouped by weekday.

As a consequence of the above findings the number of passengers, together with the flight-specific retail factor remain within the quantitative research model.

#### 4.1.9 Summary: set of current flight data

Summarizing the so far analysis of actual flight data used for research, it can be said that it provided sufficient insight to calibrate the quantitative research model and to determine potentially contributing factors towards the retail result.

Those factors include the number of passengers, number of flights, performance of individual retail areas and that of individual flights. Furthermore, possible values for turnaround times have been determined. And finally, the importance of planning towards actual operations has been demonstrated in terms of gate changes and sales result.

*[ALIGNMENT ASPECT]*

*Similar to the comments within the methodology section, it needs to be stated that a good understanding of the real business world may enable a potential IT solution to produce more meaningful output (compare, Figure 10, (3) and (5)). Therefore, the data available has been analysed to some detail and will be used for a more efficient use of the simulation environment (better parameters => less trials necessary).*

With the knowledge gained from the above and the simulation environment as introduced in the methodology chapter, further analysis has been undertaken in form of scenario simulations.

## **4.2 Scenario technique**

As introduced in the methodology chapters, it is aimed to increase the retail result through application of the tailored gate allocation algorithm. Here the so far determined contributing and constraining factors need to be considered. In order to improve the result, variations of those factors would help to determine possible influence towards the sales figures. In addition, variation of those factors would allow for representation of different business setups.

The scenario technique copes for the above and is therefore chosen to obtain more insight into the area of research. The chapters below describe the scenario settings and the results delivered by the simulation environment. Analysis of the latter, finally tries to explain the contributing factors within the business environment.

### **4.2.1 Elements describing a scenario**

Basically, a scenario consists of a flight schedule (for one or more days) to be allocated within a business setting making use of a certain airport infrastructure.

The schedule is fixed and will only be altered in case some flights could not be allocated for the time as requested. For all scenarios the time frame taken is the combined period of summer season 2006 and winter season 2006/07.

The (to-be-allocated) flight schedule is based on actual flight data, not on the seasonal planning data for the period above.



Most elements of the reference data as introduced in the methodology chapter (see Figure 41) e.g. including definitions for gates and retail areas or airline alliances further describe a scenario.

In addition to the above there is commercial data describing a scenario. These items are those as introduced with the quantitative research model ( $S_{DF}$ ,  $F_{DFR}$ ,  $F_{FLIGHT}$ ,  $F_{AREA}$ ).

Finally, there are operational parameters completing the description of a scenario. These are:

- minimum time a flight is assumed to be on a gate position
- maximum time a flight is allowed to be on a gate position
- a mandatory buffer time that needs to be elapsed before a next flight is allowed to be allocated to the same gate.

Although the above already describe a business setting (scenario) to a large extent, it still does not reflect reality. Being a model, some assumptions had to be made and certain rules had to be applied.

#### 4.2.2 Standard assumptions, rule set

A very basic assumption is that gate allocation is solely conducted for departing flights. The standard ground time is to cope for that. For example, a flight arrives as an INBOUND. This flight (the aircraft) is then either to be allocated to the gate it will depart from as OUTBOUND according to the gate allocation plan, or it would need to be towed to a remote parking stand or to a next OUTBOUND gate position.<sup>119</sup>

Another assumption is that the gate allocation plan uses both contact gates and bus gates. This means that e.g. ‘gate B26’ in the meaning of passenger boarding gate within the terminal building is associated to ‘gate B26’ in the meaning of a parking (stand) position for an aircraft. So, in case of utilization of bus gates it has been assumed that there is an existing remote stand for the aircraft of that flight. Finally, in case an INBOUND flight would remain for too long on a contact gate position, it is assumed that it is towed away to a remote stand.

The flight schedule – as an input basis for the allocation algorithm – uses the scheduled time of departure (STD) as the objective to be met in allocation. Everything is

---

<sup>119</sup> For example, this is standard procedure in Frankfurt. At Singapore Changi Airport wide-body aircraft are towed from contact gate to remote stand within 55 minutes after arrival (and back within 100 minutes from STD). Even shorter times apply for smaller aircraft types.

---

calculated backwards from STD. So, a gate becomes vacant in the allocation process at STD plus time for an operational buffer<sup>120</sup>. The combination of a flight's 'standard ground time', its 'STD' and a 'scenarios-specific buffer time' represent the time frame between 'on-block-time'<sup>121</sup> and 'STD', which would include the taxi-out time. For example, if in a scenario a flight is allocated to leave gate at STD of '13:55' and a buffer of 20 minutes is applied, that gate would become available again at '14:15'. In reality the gate would become available as many minutes earlier as the taxi-out time for that flight would be. There should be no essential distortion towards the overall allocation result.

It is assumed that too crowded retail areas are to be avoided. Therefore, in most scenarios a value of 1.5 square meters per passenger has been applied to cope for that requirement.<sup>122</sup> The computation of the available space per gate considers the gate waiting area only. That means that the space in and between shops that is (usually) outside the waiting area would be available in addition the 1.5 square meters per passenger. Thus, in cases where the gate allocation algorithm has blocked gates in a retail area due to passenger congestion, still more than the 1.5 square meters might have been available (taking entire space available into account).

---

<sup>120</sup> Wu et al. (2004) also apply a buffer to minimize system costs from operational uncertainty. Yan et al. (2001, p.415) use a buffer time to "resolve minor delays that often occur in real-time operations."

<sup>121</sup> Also referred to as 'on-chocks'.

<sup>122</sup> Compare also phase three of gate allocation algorithm.

### 4.2.3 Scenarios

With the aforementioned in mind a couple of scenarios have been defined. A first scenario tries to closely reflect the actual situation, but with desirable and realistic constraints regarding the times at a gate. This scenario is called the ‘baseline scenario’. However, as sensitivity of the different parameters was to be tested, further scenarios describe different operational settings, different commercial settings, different definition of a retail area and a different mutual exclusive use of gate pairs.

Based on the analysis of the actual flight data, the baseline scenario applied the following settings to the simulation run:

| Parameter   | Value   |
|---|---|
| Average duty free spending of a passenger ( $S_{DF}$ )                  | 1.20  |
| Factor to express relation of duty free to overall retail ( $F_{DFR}$ ) | 1.43  |
| Retail area factors: R1, R2, R3, R4, R5, R6, R7                         | 1.0   1.1   2.2   1.4   1.4   1.7   2.1   |
| Minutes at gate (minimum)   | 45  |
| Minutes at gate (maximum)   | 180   |
| Buffer time   | 20  |
| Retail area definition (gates in retail area)                           | According to Table 3  |
| Function-switch to avoid over-crowded retail areas                      | ON  |
| Number of gate pairs for exclusive use                                  | 36<br>(according to the real situation in Frankfurt at the time of observation) |

**Table 24: Definition of baseline scenario.**

#### *[ALIGNMENT ASPECT]*

*In order to produce meaningful results, it is tried to simulate the real business world situation as closely as possible (compare, Figure 10, (3), (5) and (6)). The scenario definitions cope for aspects that reflect situations found at Frankfurt Airport.*

The output of the scenario simulations is discussed after the description of scenarios.

The following other situations have been aimed to model in scenarios, too:

- A turnaround time that is ‘tuned’ for marketing purpose will be increased to average values
- A turnaround time shall cope for a minimum ground handling setup time at the position and for the factor of uncertainty in the overall time on ground (e.g. late passengers, missing equipment)
- An expedited turnaround for large aircraft
- An increased duty free spending per passenger
- The possibility to have overcrowded retail areas
- The promotion or enhancement of a specific retail area, so that its factor  $F_{AREA}$  will improve
- The situation of a homogeneous retail area performance (no differences in  $F_{AREA}$ )
- Some of the gates will be assigned to another retail area, because way finding or physical settings have changed

The resulting scenario definitions can be seen in the tables below.

Within the scenarios of ‘Group 1’ (Table 25) the baseline scenario can be found (scenario AD-S\_F11). In general, the scenarios of ‘Group 1’ use parameters that are close to Frankfurt’s situation in terms of retail area definition, the values for  $S_{DF}$ ,  $F_{DFR}$  and  $F_{AREA}$ . ‘Group 1’ scenarios are used to verify the statement (H.1) and answer the research question.

| ACTUAL (SIM_FRA $S_{DF}=1.20$ , $F_{DFR}=1.43$ ) |       |                   |                   |             |            |                      |  |     |     |     |     |     |     |     |
|--|-------|-------------------|-------------------|-------------|------------|----------------------|--|-----|-----|-----|-----|-----|-----|-----|
| Scenario ID                                      |       | Min. time at gate | Max. time at gate | Buffer time | Total time | Avoid PAX over-crowd | gate use (mutually exclusive gate pairs) | R1  | R2  | R3  | R4  | R5  | R6  | R7  |
| T  | S_F1  | 45                | 180               | 20          | 245        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| U  | S_F2  | 60                | 180               | 20          | 260        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| V  | S_F3  | 90                | 180               | 20          | 290        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| W  | S_F4  | 45                | 120               | 15          | 180        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| X  | S_F5  | 45                | 120               | 30          | 195        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| Y  | S_F6  | 60                | 120               | 15          | 195        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| Z  | S_F7  | 60                | 120               | 30          | 210        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| AA   | S_F8  | 10                | 720               | 10          | 740        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| AB   | S_F9  | 10                | 720               | 30          | 760        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| AC   | S_F10 | 10                | 720               | 30          | 760        | yes                  | 18                                       | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| AD   | S_F11 | 45                | 180               | 20          | 245        | yes                  | 36                                       | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |
| AE   | S_F12 | 90                | 180               | 20          | 290        | yes                  | 36                                       | 1.0 | 1.1 | 2.2 | 1.4 | 1.4 | 1.7 | 2.1 |

**Table 25: Scenario definitions (Group 1).**

Values for the times at gate and buffer time vary according to the different situations aimed to simulate. The column ‘total time’ just sums up the time values of the previous columns. This does not indicate for how long an aircraft actually remained at the gate, but sets the constraints. So ‘total time’ must not be mistaken as an indicator for a passengers time to stay in the gate area or retail area.

Scenarios S\_F8, S\_F9 and S\_F10 virtually eliminated all time restrictions, which basically feeds the gate allocation algorithm with the actual flight schedule ‘as-is’. Additionally, in scenario S\_F10 further 12 gate pairs have been declared mutually exclusive. Scenario S\_F11 describes S\_F1, but with a total of 36 gate pairs<sup>123</sup> that cannot be used at the same time (S\_F11 is the baseline scenario). In addition to that, scenario S\_F12 increases the minimum time at gate to 90 minutes, which implies an increase of the safety buffer for flights that filed a standard ground time of less than 90 minutes. S\_F12 also compares to S\_F3, but with 30 additional gate pairs that are not to be used at the same time.

<sup>123</sup> The following 36 gate pairs have not been allowed to be allocated at the same time: A4/A5, A11/A51, A12/A52, A13/A53, A14/A54, A15/A55, A16/A56, A17/A57, A18/A58, A19/A59, A20/A60, A21/A61, A22/A62, A23/A63, A25/A65, B1/B3, B1/B4, B2/B5, B3/B4, B6/B7, B8/B9, B9/B41, B19/B20, D40/D50, D41/D51, D42/D52, D43/D53, D44/D54, E10/E23, E10/E11, E11/E24, E12/E25, E13/E26, E21/E22, E23/E24, E25/E26.

The scenarios of ‘Group 2’ (Table 26) describe a situation different from Frankfurt with changed values for  $S_{DF}$ , and  $F_{AREA}$  and a larger variety of combinations of the various parameters, and only 6 gate pairs for mutual exclusive use.

| ACTUAL (SIM_FRA $S_{DF}$ =6.50, $F_{DFR}$ =1.43, diff. retail area factors) |                   |                   |             |            |                      |  |    |     |     |     |     |     |     |     |
|---|-------------------|-------------------|-------------|------------|----------------------|--|----|-----|-----|-----|-----|-----|-----|-----|
| Scenario ID   | Min. time at gate | Max. time at gate | Buffer time | Total time | Avoid PAX over-crowd | gate use (mutually exclusive gate pairs) | R1 | R2  | R3  | R4  | R5  | R6  | R7  |     |
| A   | S1                | 45                | 120         | 15         | 180                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| B   | S2                | 60                | 120         | 15         | 195                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| C   | S3                | 90                | 120         | 15         | 225                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| D   | S4                | 45                | 180         | 15         | 240                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| E   | S5                | 60                | 180         | 15         | 255                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| F   | S6                | 90                | 180         | 15         | 285                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| G   | S7                | 90                | 120         | 15         | 225                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| H   | S8                | 45                | 150         | 30         | 225                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| I   | S1_1              | 45                | 120         | 30         | 195                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| J   | S2_1              | 60                | 120         | 30         | 210                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| K   | S3_1              | 90                | 120         | 30         | 240                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| L   | S9_1              | 60                | 180         | 20         | 260                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| M   | S9_2              | 60                | 180         | 20         | 260                  | yes                                      | 6  | 1.5 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| N   | S1_2              | 45                | 120         | 15         | 180                  | yes                                      | 6  | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| O   | S2_2              | 60                | 120         | 15         | 195                  | yes                                      | 6  | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| P   | S3_2              | 90                | 120         | 15         | 225                  | yes                                      | 6  | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| Q   | S1_3              | 45                | 120         | 15         | 180                  | no                                       | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| R   | S2_3              | 60                | 120         | 15         | 195                  | no                                       | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |
| S   | S3_3              | 90                | 120         | 15         | 225                  | no                                       | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7 | 2.1 |

Table 26: Scenario definitions (Group 2).

A third group of scenarios (not shown in a separate table) simulate the same situations as given within ‘Group 2’, but with a different definition of retail area R2 and R3. In that group R3 comprises also the gates B11, B12, B13 and B19, which belonged to R2 in ‘Group 2’ scenarios. This group of scenarios has been designed to show the effect, which e.g. an enhancement of the retail environment for certain gates may have.

---

So, a total of 50 scenarios have been defined<sup>124</sup> and will be computed in order to determine whether there might be a possible improvement compared to the situation of actual sales and to determine contributing factors.

### 4.3 Simulation runs

Using the simulation workbench as introduced in Chapter 3.7, except for two (see above), all scenarios have been run on the same computer – usually three scenarios concurrently. Depending on the parameter settings a single run<sup>125</sup> lasted between 4 and 14.5 hours. On a quad-core processor each instance of the software consumed 25% (i.e. one core) of processing power.

For each scenario the output generated consisted of a gate allocation plan and a schedule for each day simulated. After a simulation run, the corresponding daily schedules have been combined into an annual schedule used for summary reporting.

So, in total 18,200 gate allocation plans (each with 44,064 data items), 18,200 daily schedules (each with approx. 23,400 data items), 50 annual schedules (each with 8,947,770 data items) and 50 summary reports (each with 384 data items) have been generated by the simulation runs.<sup>126</sup>

Unfortunately, due to the large format of an allocation plan, it is not possible to fit into this paper piece of work (similar with schedules and summary report).<sup>127</sup> However, in order to get an idea about a gate allocation plan Figure 69 provides an excerpt.

---

<sup>124</sup> Scenarios C-S3 and G-S7 (Group 2) have been run on different machines to test timing and the deterministic character of the algorithm.

<sup>125</sup> Not split and not run concurrently in itself, but run concurrently with simulation of other scenarios.

<sup>126</sup> In addition to that a ‘cross-check-scenario’ has been simulated to compare explicitly the time period for which both retail data and actual operational data was available. This scenario is based on the baseline-scenario.

<sup>127</sup> Compressed samples are provided in Appendix B, Chapters 9.3 to 9.6.

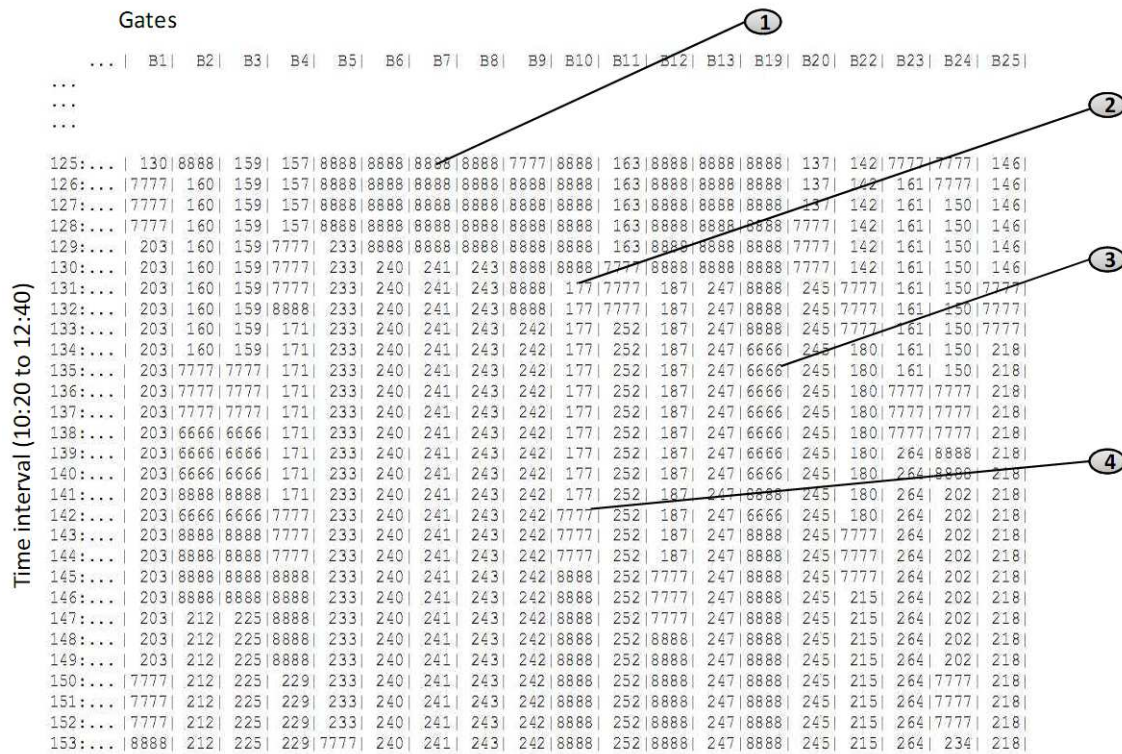


Figure 69: Excerpt of a sample gate allocation as procured by the simulation workbench.

In general, the above is to be read gate per gate, from top to bottom. The figures at the intersection of gate and time indicate the internal index of a flight. The time a gate is not occupied is indicated by ‘8888’ (1). The situation of (2) shows a flight (#177) that remains for 55 minutes at the gate (11 time intervals). The gate is blocked an additional 15 minutes (4) by a buffer (indicated by ‘7777’). As mentioned before, a situation may arise where a retail area may become too crowded. In those cases the algorithm blocks any gate in that retail area for the next time interval until the situation will have improved again. This is indicated by ‘6666’ in (3).

Except for scenario S\_F12 (Group 1), during none of the simulation runs a flight has been unable to be allocated in the time interval as requested. Internally, the workbench computed the following two additional values for each simulation run:

1. A sort of opportunity cost has been computed, indicating how much additional sales might have been generated in case enough gates in the preferred retail areas would have been available. The values ranged between 2.46% and 7.58% (4.35% average).
2. Another sort of opportunity cost has been computed, indicating how much sales have not been generated because not the best trial in the solution stack could be



allocated successfully. The values ranged between 0.10% and 0.19% (0.15% average). In other words, the algorithm achieved in average 99.85% of the possible retail sales under the constraints given.

The massive amount of simulation data have basically been aggregated in the 50 summary reports and transferred into a spreadsheet for further analysis.

#### **4.4 Analysis of results**

The following analysis is basically based on quantitative measures. In the analysis the qualitative aspect of gate allocations is not looked at in detail. Nevertheless, *basic quality aspects* have been satisfied by means of the simulation parameters as introduced above.

In order to obtain more insight into the individual contributors of an improved retail result, a more differentiated view is provided below.

#### 4.4.1 Result of baseline scenario compared to actual result

The overall outcome of the baseline scenario is an increase in sales of approx. 17% (i.e. approx. 13.5 million Euros). This was achieved by a re-distribution of flights (and consequently passengers) across the retail areas. Figure 70 shows this on an average daily basis.

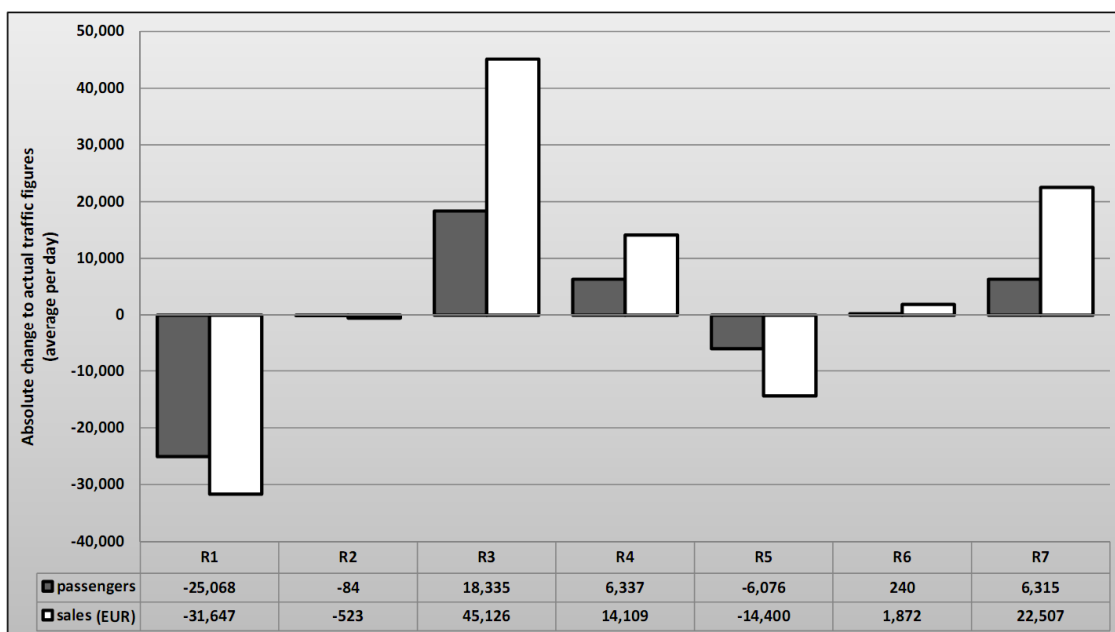


Figure 70: Comparison of actual figures to those of the baseline scenario.

From relatively low performing R1 approx. 85% of passengers have been shifted towards R3 and R4.<sup>128</sup> And within Terminal 2 almost all flights from gates associated with R5 have been shifted to R7.

As expected, the amount of passengers per day remained the same<sup>129</sup>. Average increase of sales per day was evenly distributed across each day of week (approx. 17%). So, the potential in increased sales was not dependent on the day of week.

The observed loss of sales in R1 and R5 is more than compensated by increased sales in the remaining retail areas. For example, one precondition that can be derived from the above figures is that the (passengers') way to the gates of R3 would need to be able to cope for more than 18,000 additional passengers per day. This means that e.g. all check-

<sup>128</sup> As gates in R3 are for transit and non-EU flights, passengers that depart from R3 would need to accept border control checks unless other ways are found to address that issue. In a real world situation this may constitute a major challenge to overcome, because sufficient inspection capacity needs to be arranged for.

<sup>129</sup> The algorithm does not shift across the change of day.

points (e.g. security, outbound border control) would need to provide sufficient capacity for those passenger flows.

Detailed figures of this comparison, both on passengers and sales can be found in Appendix B, Chapters 9.10 and 9.11.

### 4.4.2 Scenario results in order of retail sales result

In order to get an idea of the most enhancing (or limiting) factors a list of all scenarios (and respective results) is provided below.

| ACTUAL (SIM_FRA $S_{DF}=1.20$ , $F_{DFR}=1.43$ ) |       |                   |                   |             |            |                      |  |     |     |     | 78,595,197 | 0.00 |     |     |            |                   |
|--|-------|-------------------|-------------------|-------------|------------|----------------------|--|-----|-----|-----|------------|------|-----|-----|------------|-------------------|
| Scenario ID                                      |       | Min. time at gate | Max. time at gate | Buffer time | Total time | Avoid PAX over-crowd | gate use (mutually exclusive gate pairs) | R1  | R2  | R3  | R4         | R5   | R6  | R7  | Sales      | % diff. to actual |
| AA   | S_F8  | 10                | 720               | 10          | 740        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 95,218,816 | 21.15             |
| W  | S_F4  | 45                | 120               | 15          | 180        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 94,703,824 | 20.50             |
| Y  | S_F6  | 60                | 120               | 15          | 195        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 92,230,637 | 17.35             |
| T  | S_F1  | 45                | 180               | 20          | 245        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 92,219,867 | 17.34             |
| AD   | S_F11 | 45                | 180               | 20          | 245        | yes                  | 36                                       | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 92,078,966 | 17.16             |
| X  | S_F5  | 45                | 120               | 30          | 195        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 91,537,586 | 16.47             |
| AC   | S_F10 | 10                | 720               | 30          | 760        | yes                  | 18                                       | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 91,002,905 | 15.79             |
| AB   | S_F9  | 10                | 720               | 30          | 760        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 90,996,345 | 15.78             |
| U  | S_F2  | 60                | 180               | 20          | 260        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 90,375,283 | 14.99             |
| AE   | S_F12 | 90                | 180               | 20          | 290        | yes                  | 36                                       | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 89,785,685 | 14.24             |
| Z  | S_F7  | 60                | 120               | 30          | 210        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 89,352,807 | 13.69             |
| V  | S_F3  | 90                | 180               | 20          | 290        | yes                  | 6  | 1.0 | 1.1 | 2.2 | 1.4        | 1.4  | 1.7 | 2.1 | 86,042,169 | 9.48              |

Table 27: Scenario results (Group 1), sorted by sales result.

Within ‘Group 1’ scenarios, the different operational times at a gate have been altered. There is no direct relationship between the *total* time (sum of individual time parameters) and the sales result to be observed. Nevertheless, it can be observed that the higher the value for the minimum time at gate (remaining parameters unchanged) the less increase in potential sales was achieved (W-Y; X-Z; T-U-V). The difference observed here was approx. 3% less sales increase per 15 minutes additional time at the gate. According to the frequency of occurrence of standard ground times (see again Table 21) this could be expected. Any increase above 45 minutes will lead to an increase of the standard ground time during simulation runs. Therefore, the gate resources will be occupied for a longer period and there is less possibility to allocate flights to retail-favoured gates. The potential solution space is reduced. The same can be observed with the buffer time (W-X; Y-Z).

From an operations perspective it has been observed that only in *AE* (scenario F\_12, Group 1) the alliance rule had to be disobeyed in *several* cases. On 332 out of 364 days in one specific time interval per day it has not been possible to allocate flights in accordance with that rule. Regarding ‘*gate use*’ it needs to be mentioned that in cases where there are high restrictions due to minimum or maximum time at gate or buffer time, a high number of mutually exclusive gate pairs lead to re-allocation of flights that are then allowed to disobey the alliance rule. This in consequence may lead to increased retail sales (in case an alliance rule would have forced a flight to be allocated in a retail area with a lower  $F_{AREA}$ , compared to allocation without alliance rule). This effect can be observed in scenarios (*AE-V*; *AC-AB*). The scenario pair (*T-AD*) performed as expected, because neither of minimum time or maximum time at gate have constrained the allocation algorithm to an extent that the alliance rule had to be disobeyed. Thus, in this case flights have only occasionally been allocated to gates in retail areas with less sales potential (but: no allocation into a ‘non-alliance-compliant’, higher  $F_{AREA}$ ) retail area. For clarification and in reference to the above discussion regarding standard ground times, it is re-stated that it really depends on the data in the flight plan (which feeds the simulation runs) whether or not e.g. maximum time at gate influences the sales result. The more the value for ‘maximum time at gate’ forces a flight to free a gate, the higher the potential for increased retail sales.

Returning to the observed violations of the alliance rule it needs to be mentioned that the spread across the day of week is quite even (Table 28). However, looking at those time intervals that caused to violate the alliance rule most often, the distribution is not as even. For example, the Wednesdays seem to have a flight schedule that does not cause problems at specific times, but violations are spread all over the day (49 observations, but only 14 amongst the top contributing intervals).

|   |                      | MON | TUE | WED | THU | FRI | SAT | SUN |
|---|----------------------|-----|-----|-----|-----|-----|-----|-----|
| Number of times rule disobeyed (both seasons) |                      | 49  | 45  | 49  | 47  | 48  | 49  | 45  |
| Most violating intervals                      | Interval 134 (11:05) | 22  | 12  | 5   | 28  | 14  | 24  | 19  |
|   | Interval 91 (07:30)  | 0   | 14  | 0   | 0   | 15  | 3   | 1   |
|   | Interval 104 (08:35) | 0   | 0   | 0   | 0   | 0   | 7   | 14  |
|   | Interval 99 (08:10)  | 4   | 0   | 0   | 9   | 8   | 0   | 0   |
|   | Interval 136 (11:15) | 3   | 2   | 9   | 0   | 0   | 0   | 1   |

Table 28: Violations of alliance rule per day of week.

Violations of the alliance rule are due to dense traffic at those times. They occur usually during the four traffic waves (compare Figure 54). Nevertheless, as indicated above and

shown in Table 29 the overview of frequency distribution of occurrences may provide input for flight planners. For example, flights may be slightly shifted (e.g. +/- five minutes) in order to de-peak the situation.

| Number of days rule disobeyed     | 124 | 33 | 21         | 11-16                   | 5-10                                      | 1-4  |
|-----------------------------------|-----|----|------------|-------------------------|---|--|
| In the following time interval(s) | 134 | 91 | 104;<br>99 | 136;<br>108;<br>106; 92 | 204; 203;<br>202; 138;<br>137; 107;<br>98 | 259; 258; 250; 248; 246; 205;<br>195; 151; 149; 148; 139; 115;<br>114; 112; 111; 110; 109; 105;<br>103; 102; 100 |

**Table 29: Violations of alliance rule: frequency classes.**

*AE* (scenario F\_12, Group 1) compared to *AD* (scenario F\_11, Group 1) shows the only difference in an increase of the value for ‘minimum time at gate’ from 45 to 90 minutes. This means that each flight is given at least 90 minutes at a gate regardless what has been filed in the schedule. In case a flight arrives late or would not yet have been towed to a gate, the gate would already have been reserved for that flight. This of course incurs a considerable negative impact on gate allocation planning. Nonetheless, it copes for the factor of uncertainty in operations (the same as with the buffer time) and should decrease (short-term) operational delays.

Despite this burden simulation results still lead to increased sales of approx. 14% compared to actual sales (and no 100% compliance with alliance rule).

Observations in ‘Group 2’ and ‘Group 3’ show similar results for the maximum time at gate (reciprocally, because the lower that time the larger the solution space would be). According to Table 21 approx. 22,000 flights would stay an hour less at a gate in case the maximum gate time would be reduced from 180 to 120 minutes. This increase in solution space can also be seen in a better sales result (*W-T*, Group 1).<sup>130</sup>

Nevertheless, a change of one (or more) time parameters is no guarantee for an improved result. Although the solution space may be increased, the structure of the flight schedule may outweigh this in a form that at certain times of a day (time intervals in the algorithm) a later time at gate (because of reduced maximum time at gate) may cause congestion in that time interval. This in consequence reduces (not increases) solution space and sales may be less.

<sup>130</sup> Although here buffer time is also reduced by five minutes. But results from ‘Group 2’ and ‘Group 3’ support this as well.

So for an overall optimum it is important to first de-peak the flight schedule (small shifts may be sufficient) and then start allocation planning, or to allow the gate allocation algorithm to vary e.g. buffer time by 5 to 10 minutes.

The results of ‘Group 2’ (Table 30) provide more insight towards potential contribution of individual parameters.

| ACTUAL (SIM_FRA $S_{DF}$ =6.50, $F_{DFR}$ =1.43, diff. retail area factors) |      |                   |                   |             |            |                      |  |     |     |     |     |     | 409,251,492 | 0.00 |             |                   |
|---|------|-------------------|-------------------|-------------|------------|----------------------|--|-----|-----|-----|-----|-----|-------------|------|-------------|-------------------|
| Scenario ID   |      | Min. time at gate | Max. time at gate | Buffer time | Total time | Avoid PAX over-crowd | gate use (mutually exclusive gate pairs) | R1  | R2  | R3  | R4  | R5  | R6          | R7   | Sales       | % diff. to actual |
| N   | S1_2 | 45                | 120               | 15          | 180        | yes                  | 6  | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0         | 2.0  | 507,911,119 | 24.11             |
| O   | S2_2 | 60                | 120               | 15          | 195        | yes                  | 6  | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0         | 2.0  | 507,911,119 | 24.11             |
| P   | S3_2 | 90                | 120               | 15          | 225        | yes                  | 6  | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0         | 2.0  | 507,911,119 | 24.11             |
| Q   | S1_3 | 45                | 120               | 15          | 180        | no                   | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 489,539,966 | 19.62             |
| A   | S1   | 45                | 120               | 15          | 180        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 489,539,521 | 19.62             |
| D   | S4   | 45                | 180               | 15          | 240        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 485,314,612 | 18.59             |
| R   | S2_3 | 60                | 120               | 15          | 225        | no                   | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 480,769,007 | 17.48             |
| B   | S2   | 60                | 120               | 15          | 195        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 480,604,354 | 17.43             |
| M   | S9_2 | 60                | 180               | 20          | 260        | yes                  | 6  | 1.5 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 480,342,294 | 17.37             |
| I   | S1_1 | 45                | 120               | 30          | 195        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 476,993,152 | 16.55             |
| E   | S5   | 60                | 180               | 15          | 255        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 475,999,672 | 16.31             |
| L   | S9_1 | 60                | 180               | 20          | 260        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 473,363,213 | 15.67             |
| H   | S8   | 45                | 150               | 30          | 225        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 473,197,033 | 15.62             |
| J   | S2_1 | 60                | 120               | 30          | 210        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 469,169,730 | 14.64             |
| S   | S3_3 | 90                | 120               | 15          | 225        | no                   | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 460,690,285 | 12.57             |
| C   | S3   | 90                | 120               | 15          | 225        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 460,678,475 | 12.57             |
| G   | S7   | 90                | 120               | 15          | 225        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 460,678,475 | 12.57             |
| F   | S6   | 90                | 180               | 15          | 285        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 457,201,756 | 11.72             |
| K   | S3_1 | 90                | 120               | 30          | 240        | yes                  | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1  | 450,455,993 | 10.07             |

Table 30: Scenario results (Group 2), sorted by sales result.

As expected an improved performance of the individual retail areas led directly to an increase in sales. And of course, as long as a flight could be allocated there would be no difference between different scenarios (*N-O-P*). In such a case the allocation rules cause the algorithm to distribute flights evenly across all retail areas, always starting with the first available gate, and as each retail area has the same  $F_{AREA}$  (2.0), it leads to same retail sales<sup>131</sup>. Once more it has to be mentioned that the columns ‘minimum time’, ‘maximum time’ or ‘total time’ are not related to a passenger’s dwell time, but to the aircraft. In fact, they limit the algorithm’s capability to determine favourable gates.

The constraint to avoid over-crowded retail areas barely had an effect on the result (*Q-A*; *S-C*; *R-B*). In general, a different number of ‘6666’ blockages does not necessarily lead to a different retail result. In fact it is the structure of the flight schedule in

<sup>131</sup> Compare precondition for research subject in Chapter 1.3, paragraph ‘Airports’.

---

conjunction with the time(s) at gate and the buffer time that determine the ‘net gate supply’ in each single time interval. So, in case an ‘over-crowded-situation’ occurs at a time when there are eligible gates in a retail area with a similar  $F_{AREA}$ , almost nothing changes. But in case the only eligible gates would have been detected in a retail area with a lower  $F_{AREA}$ , then the retail result would have decreased accordingly.

Comparing scenarios (*L*) and (*M*) delivers the expected higher result for (*M*), because of its higher  $F_{AREA}$  value for R1. Compared to scenario (*L*) in scenario (*M*) more flights (and resulting passengers) have been allocated at R1, so that no flight needed to be allocated in R4. And as expected (due to allocation rule set), almost nothing changed in R5, R6, R7.<sup>132</sup>

For the reason of clarification it is mentioned again that the total time column does not relate by any means to a passenger’s duration in retail areas, and thus does not *directly* imply anything regarding the retail result.

As introduced with the scenario description (see Chapter 4.2.3) the third group has a different definition of retail areas *in terms of the gates they comprise*. Everything else remains unchanged.

---

<sup>132</sup> Details can be found in form of an excerpt from a summary report produced by the simulation workbench in Appendix B).

As observed in Table 31 the overall result of ‘Group 3’ scenarios is similar to that of ‘Group 2’. Due to the fact that the attribution of the four gates changed from R2 in ‘Group 2’ to R3 in ‘Group 3’, more gates had the possibility to perform with an increased factor (2.0 instead of 1.5). In consequence, a slightly overall increased sales result could be achieved in this group.

| ACTUAL (SIM_FRA $S_{DF}=6.50$ , $F_{DFR}=1.43$ , R2 and R3 diff. gates, diff. retail area factors) |                   |                   |             |            |                      |  |  |     |     |     |     |     | 412,590,152 | 0.00  |                   |       |
|--|-------------------|-------------------|-------------|------------|----------------------|--|--|-----|-----|-----|-----|-----|-------------|-------|-------------------|-------|
| Scenario ID  | Min. time at gate | Max. time at gate | Buffer time | Total time | Avoid PAX over-crowd | gate use (mutually exclusive gate pairs) | Diff. to 'group two', gates R2->R3: B11, B12, B13, B19 |     |     |     |     |     |             | Sales | % diff. to actual |       |
|  |                   |                   |             |            |                      |  | R1   | R2  | R3  | R4  | R5  | R6  | R7          |       |                   |       |
| N  | S1_2              | 45                | 120         | 15         | 180                  | yes                                      | 6  | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0         | 2.0   | 507,911,119       | 23.10 |
| O  | S2_2              | 60                | 120         | 15         | 195                  | yes                                      | 6  | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0         | 2.0   | 507,911,119       | 23.10 |
| P  | S3_2              | 90                | 120         | 15         | 225                  | yes                                      | 6  | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0         | 2.0   | 507,911,119       | 23.10 |
| Q  | S1_3              | 45                | 120         | 15         | 180                  | no                                       | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 495,183,849       | 20.02 |
| A  | S1                | 45                | 120         | 15         | 180                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 491,251,290       | 19.07 |
| D  | S4                | 45                | 180         | 15         | 240                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 486,801,694       | 17.99 |
| R  | S2_3              | 60                | 120         | 15         | 195                  | no                                       | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 486,797,208       | 17.99 |
| M  | S9_2              | 60                | 180         | 20         | 260                  | yes                                      | 6  | 1.5 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 485,573,233       | 17.69 |
| B  | S2                | 60                | 120         | 15         | 195                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 483,319,805       | 17.14 |
| I  | S1_1              | 45                | 120         | 30         | 195                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 481,714,871       | 16.75 |
| E  | S5                | 60                | 180         | 15         | 255                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 479,052,025       | 16.11 |
| H  | S8                | 45                | 150         | 30         | 225                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 478,927,961       | 16.08 |
| L  | S9_1              | 60                | 180         | 20         | 260                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 476,792,980       | 15.56 |
| J  | S2_1              | 60                | 120         | 30         | 210                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 473,668,548       | 14.80 |
| S  | S3_3              | 90                | 120         | 15         | 225                  | no                                       | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 466,842,101       | 13.15 |
| C  | S3                | 90                | 120         | 15         | 225                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 465,114,845       | 12.73 |
| G  | S7                | 90                | 120         | 15         | 225                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 465,114,845       | 12.73 |
| F  | S6                | 90                | 180         | 15         | 285                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 461,386,728       | 11.83 |
| K  | S3_1              | 90                | 120         | 30         | 240                  | yes                                      | 6  | 1.0 | 1.5 | 2.0 | 1.4 | 1.4 | 1.7         | 2.1   | 454,606,966       | 10.18 |

Table 31: Scenario results (Group 3).

So, the (retail-wise) enhancement or different utilization of gates directly resulted in an improved sales result. Nothing else is different in ‘Group 3’ scenarios. The slight changes in order result from the changed attribution of gates to the retail areas R2 and R3.

Therefore, any explanation given for Table 30 also applies to Table 31.

The **elements that contributed most** regarding the shift of passengers were (a) the retail area factors and (b) the rule set. The latter has basically been responsible for a separation between Terminal 1 and Terminal 2. As it would have been too far apart from real conditions no simulations without the basic rule set have been conducted.<sup>133</sup>

<sup>133</sup> Constraint relaxation has only been necessary in scenario F\_12, Group 1.



***Best results could be achieved*** with those scenarios not constraining gate allocation by too much time reserves and with a maximum number of gates for concurrent use.

In some scenarios it could be observed that almost the complete traffic via a certain retail area was distributed to other retail areas, so that theoretically the gates would not have been allocated during the entire season at all.

According to the simulations of 'Group 1' the ***financial benefit*** would have been between 7.4 million and 16.6 million Euros.<sup>134</sup> However, these figures would need to be set in relation to the additional cost at operations, and they need to be looked at along with the assumptions that have been taken for granted in each scenario.

Finally, a cross-check has been run with a scenario comparing only the period of time for which both retail data and actual operational data has been available (2007-01-01 to 2007-03-24; i.e. 50,175 flight events ). The retail area factors taken were those from the winter season (compare Table 14). Everything else has been taken as defined in the baseline scenario (Group 1, S\_F11). The results show an increase in sales revenue from approx. 16.4 million Euros to 19.8 million Euros (i.e. almost 21%).

The ***overall results*** strongly suggest that under the constraints given there is considerable potential to increase retail sales compared to actual sales (latter with no retail focus in the gate allocation process). The achieved values range from approx. 9% to approx. 21% increased sales across the scenarios from Group 1.

***This assesses the statement (H.1) to be feasible and answers the research question raised in Chapter 2.5 to be true.***

Despite the potential drawbacks because of assumptions and available data, there is still a positive statement regarding a possible application of the above. However, applying the above at different airports may alter some of those assumptions and may result in a different outcome.

---

<sup>134</sup> Keeping in mind that only a proportion thereof will be obtained by the airport operator, depending on the contracts with the retailers.

*[ALIGNMENT ASPECT]*

*Having compared a process output with and without additional support of the IT function (process integration on data layer and on business rules layer, and IT application for gate allocation planning task), a to some extent positive contribution has been determined. So, the example – derived from application of the information intensity concept – seems to support the assumption of positive impact of business/IT alignment on a business' output.*

The main conclusions, a guideline for application at other airports, limitations of the approach undertaken as well as further areas of research are discussed in the concluding chapter.

*5. CONCLUSIONS  
AND  
RECOMMENDATIONS  
FOR FURTHER RESEARCH*



## 5. CONCLUSION AND RECOMMENDATIONS FOR FURTHER RESEARCH

Having applied the research methodology to a specific business context, the results achieved are promising. In addition to the achievement regarding the statement (H.1) (see previous chapter), the aim and objectives have been achieved as follows:

(AIM) *To appraise in how far<sup>(A)</sup> the use of information technology<sup>(B)</sup> may foster a core airport operational process (i.e. gate allocation) taking into account the requirements of an airport commercial process (i.e. airport retailing).*

- (A) Extent: Approx. percentage change in retail revenue (results in Chapter 4)  
 (B) Use of IT:
- IT-supported process decomposition (Chapters 3.2.3 and 3.2.4)
  - IT-supported data modelling (Chapter 3.2.5)
  - IT-supported simulation of business rules to produce operationally feasible allocation plans (Chapters 3.4.4, 3.7, 4.2, 4.3)
  - IT-supported monetary evaluation of produced allocation plans (Chapter 4.3)

(O.1) *To determine the limits of general purpose methods to model business alignment of IT<sup>(A)</sup>, and placement of the research topic within existing frameworks.<sup>(B)</sup>*

- (A) Limits
- Lack of causality (IT as input; monetary figures as output); compare discussion in Chapter 2.2 and the resulting application of the information intensity model.
  - Assumption that there would be appropriate IT support available. This has been addressed by actual development of a tailored software application taking into account constraints of IT architecture and the element of cost.
- (B) Placement
- Summarizing the discussion in Chapter 2.2, Figure 10 places the research topic within existing frameworks.

(O.2) *To develop a possible extension or a more specific application of existing methods<sup>(A)</sup> in a way that the derived methodology may be used for further processes in the airport business environment.<sup>(B)</sup>*

- (A) Application
- Partly application of the concept of information intensity by modelling an airports value chain (Figure 18) and placement of airport processes in the information intensity matrix (Figure 19).
- (B) Further processes in airport environment
- Basically all of the processes in information intensity matrix may be a candidate for IT support. The latter would need to be derived from a detailed analysis of the processes chosen. As suggested, methods for analysis may be process decomposition and information modelling.

(O.3) *To construct a conceptual model<sup>(A)</sup> describing the relationship between airport retailing and gate allocation. The model should be based on process structure (in form of a business process decomposition)<sup>(B)</sup>, as well as on information structure (in form of a data model)<sup>(C)</sup>. Furthermore, it should be quantifiable for later simulation purposes.<sup>(D)</sup>*

- (A) A conceptual model has been presented in Chapter 3.3.  
 (B) A business process decomposition has been conducted in Chapters 3.2.3 and 3.2.4.  
 (C) A data model has been developed in Chapter 3.2.5.  
 (D) A quantified model has been developed in Chapter 3.5.

(O.4) *To develop an algorithm for the gate allocation process that copes for the needs of supporting the retailing process (i.e. an increased sales result).*

→ A problem-tailored algorithm has been developed in Chapter 3.6.3.

(O.5) *To develop an independent simulation environment with implementation of the algorithm as outlined in O.4.*

→ The simulation environment has been developed. Chapter 3.7 describes its basic elements.

In summary the following alignment aspects have been addressed<sup>135</sup>:

- *IT support is aligned with industry (compare, Figure 10, (1)).*
- *IT support is aligned with own value chain (compare, Figure 10, (2)).*
- *IT support is aligned with own business processes (compare, Figure 10, (2)).*
- *IT support is aligned with own information flows (compare, Figure 10, (2)).*
- *IT support is aligned with business function, because it understands own business (compare, Figure 10, (2) and (3)).*
- *IT support is aligned with business as the IT function considers IT investments to increase business output or to decrease cost, and in line with budgets (compare, Figure 10, (4)).*
- *IT support is aligned with business as requirements of IT architecture are considered in the IT solution (here hardware and software requirements) (compare, Figure 10, (5)).*
- *IT support is aligned with business as the change in business output (to some extent) can be measured (compare, Figure 10, (6)).*

Transferring the knowledge gained from the research work, a couple of action items have been derived and are proposed to airport managers in the following chapter.

## 5.1 Recommended actions for airport managers

Simulation results suggest that for a possible increase in sales, the following aspects are recommended for immediate consideration:

1. Integrate the ‘retail-orientation objective’ in the seasonal flight planning process.
2. Identify the most contributing flights requiring the least change in current operational setting. Initiate re-allocation of those flights.

<sup>135</sup> Compare also the links to alignment, marked as ‘[ALIGNMENT ASPECT]’ throughout the thesis.

3. Depending on the level of automation in the gate allocation process and on the IT systems used to support the allocation planning task, a corresponding rule set shall be complemented by rules that allow for a retail-orientation.
4. In order to increase awareness, add a 'retail revenue tag' to each allocation plan.

For further consideration the following is suggested:

5. Consider elements that allow most flexible use in regard to investment in terminal infrastructure.
6. Appraise any procedure that constrains a free gate allocation regarding the potential opportunity cost incurred (because of possibly less retail sales). Foster the application of such a measure to be used into any associated discussion (e.g. allocation wishes of airlines, or limited use of infrastructure due to implementation of security rules based on political decisions).
7. Depending on the level of integrated resource planning (check-in counters, gates, stands, security check points, baggage belts, etc.) incorporate findings into overall planning model.

For a minimum, there should be an awareness regarding the sales aspect in the gate allocation planning process. This ought to be present in any corresponding discussion as mentioned above in point 6.

*At Frankfurt Airport* – as a direct result from having presented first findings of this research – the seasonal flight planning department have picked up the idea of a 'revenue tag' in conjunction with any planned allocation (compare above). A prototype software solution has been developed to add such a tag to any allocation plan. Furthermore, flight plan data has been analysed for candidate flights to be re-allocated for a potentially more retail-favoured gate. In case there is the possibility to do so, those flights will be re-allocated and corresponding changes in actual retail revenue will be analysed. And finally, the magnitude of the individual retail area factors ( $F_{AREA}$ ) are being analysed for qualitative contributors. This would allow for a more detailed model in respect to  $F_{AREA}$ . In summary, Frankfurt's awareness regarding the potential embedded in combining gate allocation in seasonal flight planning and retail business has been increased considerably.

However, having taken the processes of gate allocation and retailing as an example to demonstrate possible contribution from the IT function, this aspect is summarized below on a more strategic level.

## **5.2 The role of IT: Strategic implications for airport policy**

The discussion so far has started with a broad view on the contribution of information (technology) within the airport business environment, and has then become very specific to produce a quasi quantitative result.

From a more strategic view (in addition to implications already mentioned above) an airport should be aware of the financial potential that can be realized upon consequent analysis and harmonization of the most information intensive processes in its business environment. In some situations such an analysis may even lead to new products or services to be offered.

However, a prerequisite would be to have a repository of processes and information flows for the entire business environment. Usually, such a repository is not available at all or only for individual players in the business context. Nevertheless, at least an integrated high level view on processes and information flows would be necessary to gain any advantage. This implies that all partners at an airport should be encouraged to contribute towards such an integral view. In most cases it will be the airport operator or the major airline at an airport that already has a majority of processes and information entities documented in a form to be usable in the above context.

So, in the business units, roles in charge of

- information quality assurance (incl. data quality and business intelligence),
- business process engineering,
- new service development, product development,

along with the enabling counterparts of the IT department should establish regular communication with the corresponding roles of their partners in the airport environment. On a more formal basis such an information exchange may be fostered via appropriate sections in an airport's user agreement. Such an agreement may require any user at the airport to submit certain pieces of information to (e.g.) the airport's IT department.

However, owing to the nature of the airport industry, each business setting at an airport is to a large extent unique. Therefore, an application of research methodology and simulation workbench for other airports needs to consider certain aspects to be discussed below.

### **5.3 Transferability**

In general, the methodology and scenarios can be applied to other airports as well.

As outlined in the previous chapters, there are certain preconditions and assumptions that have to be met in order to obtain simulation results as described before.

However, exactly those constraints may limit the application of the above in respect to other airports. The following discusses different aspects to be considered when trying to apply the research methodology to a different business set up.

#### **5.3.1 Different players in same business context**

First of all, there needs to be a constellation in place allowing for a seasonal flight planning process that considers both operational and commercial aspects. As long as such a process exists, it does not matter whether a certain player (process owner) is within the same legal entity or a business partner (from another company or authority).

Usually, the airport operator, the retailer(s) and the airlines would need to support the common goal of increased retail sales. This is very similar to all initiatives around collaborative decision making (CDM) where the airport operator, the airlines and air traffic control need to agree to a single shared goal, which is to increase the performance of the overall 'system airport'. This means that there need to be incentives in place that over-compensate drawbacks from individual operational decisions.

The level of vertical or horizontal integration at an airport does not really matter, if there is a common understanding (embedded in contracts) between all players involved.

As introduced in Chapter 3.2.2 in general the research topic can be applied to airports of different type. Nevertheless, there will be a significant bandwidth in terms of potential additional sales generation.

Regardless the airport environment in which it is aimed to apply the methodology and the simulation workbench, a couple of aspects need to be considered in doing so.

In general, an application at another airport would need to be conducted as outlined in Figure 71.



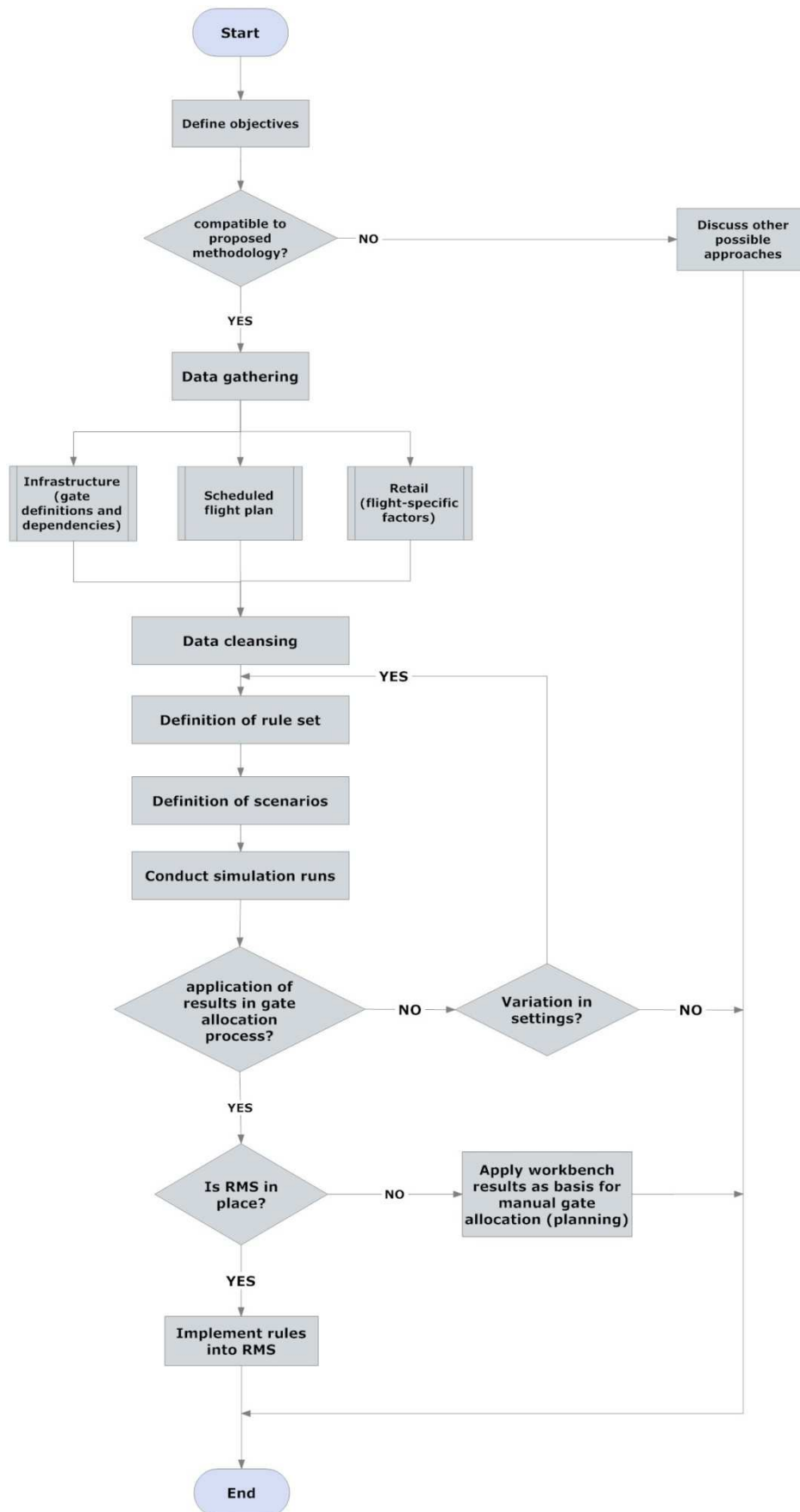


Figure 71: Application of methodology at other airports.

### 5.3.2 Individual aspects to be considered

During the *data cleansing* task it became very evident that complete and consistent data comprising *infrastructural data, flight plan data and retail data* is crucial to the overall ability to apply the research topic to other airports.

It has to be stressed that the formal requirements from methodology and especially from the simulation workbench are minimal. After data cleansing, corresponding plain text files of delimited line format are sufficient in terms of data source.

The most effort will need to be put into the *data gathering* itself and the data cleansing task. Data gathering will depend on the number of players and the level of integration amongst them.

A challenge in the data cleansing task will be to *match commercial (retail) data with standard flight plan data*. Based on the retail data (and together with retail experts from the airport in concern) the different retail areas would need to be identified and corresponding retail area factors need to be determined.

In case there were any aspects in the *infrastructural layout of an airport* not covered by the gate definition, but necessary for the overall gate allocation process, this would need to be re-programmed within the simulation workbench.

The same applies to the rule set. Constraints like aircraft size, gate size, alliance affiliation or passenger density within retail areas are considered within the simulation model. Still to be integrated would need to be e.g. capacity measures of points within the passenger (way finding) process like security check points – or in general the definition of passenger ways (e.g. via a waypoint matrix in terms of distance and time). The latter is especially important to airports performing a hub function with a large proportion of transfer passengers.<sup>136</sup>

Given the targeted heterogeneous retail setting as described in Chapter 3.2.2 the basic requirement to transfer application of the research topic to other airports would be to obtain flight-specific retail sales data. Further important information to be obtained from the retailers would be an indication regarding the maximum number of passengers to be catered for at a time. In the research model only the floor space of the gate (hold) rooms has been considered. There are more aspects to be considered when applying

---

<sup>136</sup> The number of passengers at a time in a location may also be determined using agent-based simulation methods (and tools). Such a method is usually more flexible regarding changes in infrastructure setting.

research results in practice. This is also due to the model character and scope of this piece of work and will be discussed below.

#### 5.4 Limitations of research undertaken

The author acknowledges that the research presented in this paper is limited and cannot be generalized without consideration of its assumptions and shortcomings.

Mentioned with the introduction of the conceptual research model, there are assumptions underlying the model, and there are simplifications compared to the real world. This enabled to examine the research object to a great level of detail. Given the amount of time and other resources, it would not have been possible to produce this piece of research if incorporated all aspects of the business environment.

- A major assumption within the research model is that there may incur *additional cost at operations* (e.g. ground handling services). This might be in case there would be an increased number of tows, or the turnaround of a flight would require more personnel in order to be handled within a certain (shorter) period of time. Although the scenarios defined have applied turnaround times close to actual figures, there may be side-effects not yet discovered. In order to determine the magnitude of such additional cost at operations *simulations of apron traffic and of the aircraft turnaround process would be necessary*.
- An important assumption within the allocation constraints has been the *eligibility of gates for both Schengen and non-Schengen flights*. This of course increases the solution space for the allocation task. Therefore, within an airport's infrastructure a most flexible arrangement of security check-points (all central or all de-central) and border-crossing inspections would be a necessity to harvest the full potential of the approach in this paper.
- Another aspect limiting the overall result is that it was taken for granted that the airport in question would be able to assure *hub-connectivity* regardless of the gate allocation, as long as the alliance affiliation rule is obeyed.
- Regarding the generation of retail sales it has been assumed that the *country of destination* of a flight explains to a great extent the nationality of its passengers, and this in consequence the retail spending behaviour. Other contributing factors, like shopping atmosphere, passenger dwell-time or passenger-specific shop arrangements have not been considered directly in the model. The *retail area factor* summarizes all those aspects in a single figure.

- A retail area's performance is also due to the specific flights allocated therein. So, there is a *mutual influence of flights and retail area performance*. A proportion of that performance is due the retail offering itself, but another proportion is solely due to the passengers' nationality (see above). The research model assumes that the retail area factors remain unchanged for different allocations. An adjustment would be necessary to cope for this.
- As it was not in scope of this paper, it has not explicitly looked at *alternative ways to enhance retail revenue in case gate allocation would not have been improved for retail objectives*. However, possibilities here would be to launch customer segment-specific promotions, to gain more intelligence about shopping behaviour in specific retail areas, to adapt retail offering (and shop advertising) even faster to the actual flow of passengers, just to name a few.
- Regarding the *infrastructure* it was assumed that the gates would be available throughout the year. This is of course not the case in real operations. However, it can be either addressed by definition of a minimum ('guaranteed') number of gates or by different scenarios (and simulations) for different infrastructure settings. The model and (with minor modification) the simulation workbench would be able to consider changed infrastructure.
- The *gate allocation algorithm* aims to find the best possible solution within a specific time interval. The ruling element is the flight schedule. For this reason sort of *local optima* will be produced. It is not looked at whether a slight change of the flight plan might produce better results.
- Another limitation lies in the *current implementation of the algorithm* all flights of a time interval will be shifted into the next interval in case not all of them could be allocated in the current interval. An identification of the optimum flight combination with as many flights as possible to be allocated in the current time interval with the remainder as candidate flights in the next interval would be an improved implementation of the algorithm. Furthermore, this may be complemented by some sort of priority function, so that those flights will receive certain preference in the next time interval.<sup>137</sup> However, as the above case did not occur, the results produced should be valid. Nonetheless, in case of a flight schedule being more dense, compared to the one used, it may become necessary to change the implementation (i.e. the simulation workbench software).

---

<sup>137</sup> Similar to the penalty adjustment methods in Yan and Tang (2007).

Despite the above limitations, the simulation results have shown that there is a potential to increase retail sales. Such an increase has (theoretically) become possible because of two information intensive airport processes were analysed and to some extent harmonized. This may be applied in broader scope by airport management in a way as mentioned above.

## 5.5 Contributions

Despite of its limitations the thesis aims to have contributed to the body of knowledge as follows:

- The concept of information intensity by Porter and Millar (1985) has partially been applied to the airport business for the first time.
- In a generic model two specific airport processes from different business domains have been combined in a way to show a potential increase in their output. This aimed to contribute towards understanding of the mutual dependency of processes within the airport business environment.
- A retail sales improving multi-objective gate allocation algorithm has been developed and implemented on standard PC hardware. This algorithm addressed the specific (retail-enhancement vs. gate assignments) problem for the first time.
- Furthermore, the ‘optimistic elements’ within the algorithm and the approach to reduce the solution space upfront have not been found in current body of knowledge before.
- The implementation in form of a simulation workbench (piece of software), helped to carry out a sensitivity analysis. An application (case study) with a similar large amount of real airport data to test a gate allocation algorithm has not been found, so far.

As occasionally suggested in-between, individual aspects of the research undertaken may be explored in more detail or expanded in scope.

## 5.6 Areas for future research

This paper has addressed many aspects within the research context. Nonetheless, the defined scope of work and limitations as described above suggest areas for further research.

It would complement the work in this paper to *describe the relationship between the retail area factor and a flight-specific retail factor*. As mentioned in Chapter 5.4 there should be some form of mutual influence between the two. For example, in empirical

studies the different elements contributing to the sales result of a retail area may be examined. Then specific flights may intentionally be allocated to gates associated with a different retail area. However, a very detailed observation of the operations environment in the terminal building would be necessary in order to be able to exclude any misleading influences. For example, information given upon check-in can be very influential towards the buying behaviour of passengers.<sup>138</sup> The result of such a study would help to modify the model in a way that the retail area factors would adjust within a simulation run (depending on the flights allocated within the corresponding retail area).

In addition to the above, the *approach may be complemented* by the following aspects:

- A more sophisticated rule set would allow to model real conditions to a greater level of detail (more process points in passengers' way).
- Alternatively, the concept of retail areas and the resulting objective function may be implemented into various other gate assignment models (see review in Chapter 2.3) for a more complete simulation.
- A multi-step approach may support solution finding in case of a flight (plan) schedule is too busy (or not balanced). Such an approach would first need to try to smoothen the peaks within the schedule. Only then in a second run the allocation algorithm as described in this paper would be applied.
- As mentioned in the chapter about 'limitations', a (quantitative) model describing additional cost at operations would help to set the results obtained in this piece of work a step further towards practice conditions.

Not found in current body of knowledge, but seen to be very useful is a *test bed for the various gate allocation algorithms* that exist. It has been observed that researchers in almost every case define their own testing environment for the solutions they suggest. Thus a performance comparison in terms of solution quality and computing time is not feasible. Hence, a standardized research environment that copes for various airport scenarios (different layouts, different flight schedules, etc.) would allow to better evaluate a proposed gate allocation solution method. In addition, a standard set of real airport data from airports of various categories would help to bridge the gap between academic research and application in practice.

---

<sup>138</sup> So observed in a case in Frankfurt where passengers were told at check-in that there would be no opportunity to purchase duty-free goods after security check (due to construction work on that day). According to the retail department, the sales figures for that flight have been significantly different from those of other days for the same flight.

Finally, as indicated within the strategic implications, there should be *further business processes* to be identified, analysed and if possible harmonized in a way to produce more favourable output. Figure 19 and the model of information intensity may help to identify such processes the same way as they supported to identify the potential that has been determined upon application of a retail-weighted gate allocation process. When the majority of airport-specific processes would have been analysed as suggested in this paper, a comparison at industry level<sup>139</sup> might help to appraise the potential derived from an aligned IT function more precisely.

---

<sup>139</sup> Compare Hu and Quan (2005).

## 6 List of References

- Adams, Chr., Haines, M. and McLellan, L. (2003). Key Go-to-Market Issues Lead to IT Business Strategy Success. *Gartner Dataquest - Research Brief*, ITSV-WW-DP-0530. Gartner Research, Stamford, Connecticut.
- Albers, S., Koch, B. and Ruff, Chr. (2005). Strategic alliances between airlines and airports — theoretical assessment and practical evidence. *Journal of Air Transport Management*, Vol. 11, 49–58.
- Apfel, A. (2002). The Total Value of Opportunity Approach. *Research Note*, DF-17-0235. Gartner Research, Stamford, Connecticut.
- Apfel, A. (2003a). BVIT: Frameworks and Methodologies That Work. *Research Note*, AV-19-4195. Gartner Research, Stamford, Connecticut.
- Apfel, A. (2003b). How Enterprises Can Reduce Costs Even More. *Research Note*, LE-19-6332. Gartner Research, Stamford, Connecticut.
- Apfel, A. and Smith, M. (2003). TVO Methodology: Valuing IT Investments via the Gartner Business Performance Framework. *Research Note*, R-19-1910. Gartner Research, Stamford, Connecticut.
- Appold, St. J. and Kasarda, J. (2006). The appropriate scale of US airport retail activities. *Journal of Air Transport Management*, Vol. 12 (2006) 277–287.
- Avison, D., Jones, J., Powell, Ph. and Wilson, D. (2004). Using and validating the strategic alignment model. *Journal of Strategic Information Systems*, Vol. 13 (2004), 223–246.
- Babic, O., Teodorovic, D., Tosic, V. (1984). Aircraft stand assignment to minimize walking. *Journal of Transportation Engineering*. 110, 55–66.
- Benjamin, R.I., Rockart J.F., Scott Morton, M.S. and Wyman J. (1984). Information Technology: A Strategic Opportunity. *Sloan Management Review*, Spring 1984, 3-10.
- Bihl, R. (1990). A conceptual solution to the aircraft gate assignment problem using 0,1 linear programming. *Computers & Industrial Engineering*, Vol. 19 (1-4), 280-284.
- Bloem, E.A., Blom, H.A.P. and Schaik, van F.J. (2002). Advanced Data Fusion for Airport Surveillance. *Technical Paper*, NLR-TP-2002-052. National Aerospace Laboratory, NLR.
- Bolat, A. (1999). Assigning arriving flights at an airport to the available gates. *Journal of the Operational Research Society*, Vol. 50, 23–34.
- Bolat, A. (2000). Procedures for providing robust gate assignments for arriving aircrafts. *European Journal of Operational Research*, Vol. 120, 63–80.
- Bolat, A. (2001). Models and a genetic algorithm for static aircraft gate assignment problem. *Journal of the Operational Research Society*, Vol. 52, 1107-20.
- Bonnke, J. (1999). Friendlier Airports: The Goal of Delta's Top Technology Project. *Software Magazin*, 6/99, 10.
- Bork, A. (2007). Developing a retail marketing strategy to promote both airport and retailers. *AIRPORT MANAGEMENT*, Vol. 1, NO. 4, 348–356 JULY-SEPTEMBER 2007.
- Braaksma, J.P. (1977). Reducing walking distance at existing airports. *Airport Forum*, 135–145.
- Brazile, R.P. and Swigger, K.M. (1988). GATES: An airline gate assignment and tracking expert system. *IEEE Expert*, 33-39.
- Browdy, Th. (1999). The corporate information and communications hierarchy: Technological management in modern enterprises. *Auerbach Publications*, 41-10-27. 1999.



- Buendia, M. and de Barros, A.G. (2008). Critical Update of the Post-Security Retail Space Design in Passenger Departure Terminals. *Airlines Magazine*, e-zine edition 40.
- Button, K., Lall, S., Stough, R. and Trice, M. (1999). High-technology employment and hub airports. *Journal of Air Transport Management*, Vol. 5, 53-59.
- Carr, N.G. (2003). IT Doesn't Matter. *Harvard Business Review*, May 2003, 41-49.
- Cerovic, M. (1998). Global Airport Retailing. *Business Insights Consumer Report*. Datamonitor. London.
- Cheng, Y. (1997). A knowledge-based airport gate assignment system integrated with mathematical programming. *Computers and Industrial Engineering*, Vol. 32, 837-852.
- Cheng, Y. (1998). A rule-based reactive model for the simulation of aircraft on airport gates. *Knowledge-Based Systems*, Vol. 10 (1998) 225-236.
- Clark, L. and Lee, M. (2002). Applying Performance Metrics and Benchmarking to the IT Services Organization (Executive Summary). *Research Note*, ITSV-WW-EX-0130. Gartner Research, Stamford, Connecticut.
- Clarke, R. (1994). *The Path of Development of Strategic Information Systems Theory*. INTERNET RESOURCE: <http://www.anu.edu.au/people/Roger.Clarke/SOS/StratISTh.html>, retrieved on 24 April 2004.
- Cline, M., K. and Guynes, St. C. (2004). IT Investment Is Strategic to a Firm's Survival. *Information Strategy: The Executive's Journal*, Spring 2004, 39-47.
- Cohn, M.D. (2003). Service Management for Information Technology: The Evolving Role of IT Financial Management. *Auerbach Publications*, 10/03, 42-10-25.
- Correia, A.R., Wirasinghe, S.C. and de Barros, A.G. (2008). A global index for level of service evaluation at airport passenger terminals. *Transportation Research Part E*, Vol. 44 (2008) 607-620.
- Crawford, G., Melewar, T.C. (2003). The importance of impulse purchasing behaviour in the international airport environment. *Journal of Consumer Behaviour*, Vol. 3, Issue 1, pp. 85-98.
- D'Souza, D. and Mukherjee, D. (2004). Overcoming the Challenges of Aligning IT with Business. *Information Strategy: The Executive's Journal*, Winter 2004, 23-31.
- Davitt, D. (2005). Catering for Change. *Travel Retailer International*. Sep/Oct2005, pp28-35.
- Ding, H., Lim, A. Rodrigues, B. and Zhu, Y. (2004). Aircraft and Gate Scheduling Optimization at Airports. *Proceedings of the 37th Hawaii International Conference on System Sciences – 2004*.
- Ding, H., Lim, A. Rodrigues, B. and Zhu, Y. (2005). The over-constrained airport gate assignment problem. *Computers & Operations Research*, Vol. 32 (2005) 1867-1880.
- Doganis, R. (1992). *The airport Business*. Routledge, London.
- Dorndorf, U., Drexl, A., Nikulin, Y., and Pesch, E. (2007) Flight gate scheduling: State-of-the-art and recent developments, *Omega - International Journal of Management Science*, Vol. 35(3), 326-334.
- Eccles, R.G. (1991). The Performance Measurement Manifesto. *Harvard Business Review*, Jan-Feb 1991, 131-137.
- Entwistle, M. (2007). Customer service and airport retail: Stimulate passenger spending. *AIRPORT MANAGEMENT*, Vol. 1, NO. 2, 151-157 JANUARY-MARCH 2007.
- Esper, T.L. and Williams, L.R. (2003). The Value of Collaborative Transportation Management (CTM): Its Relationship to CPFR and Information Technology. *Transportation Journal*, Summer 2003, 55-65.

- Fernie, J. (1995). The coming of the fourth wave: new forms of retail out-of-town development. *International Journal of Retail & Distribution Management*, Volume 23 · Number 1 · 1995 · pp. 4–11.
- Feurer, R., Chaharbaghi, K., Weber, M. and Wargin, J. (2000). Aligning Strategies, Processes, And IT: A Case Study. *Information Systems Management*, Winter 2000.
- Forster, P.W. and Regan, A.C. (2001). Electronic Integration in the Air Cargo Industry: An Information Processing Model of On-Time Performance. *Transportation Journal*, Summer 2001, 46-61.
- Fraport (2006). *Geschäftsbericht 2006*. (Annual report), Fraport AG, Frankfurt, Germany.
- Fraport (2007). *Geschäftsbericht 2007*. (Annual report), Fraport AG, Frankfurt, Germany.
- Freathy, P. and O'Connell, F. (1998a). Supply chain relationships within airport retailing. *International Journal of Physical Distribution & Logistics Management*, Vol. 28, No. 6, 1998, pp. 451-462.
- Freathy, P. and O'Connell, F. (1998b). The role of the buying function in airport retailing. *International Journal of Retail & Distribution Management*, Vol. 26, No. 6, 1998, 247-256.
- Freathy, P. and O'Connell, F. (1998c). *European Airport Retailing: Growth Strategies for the New Millennium*. Macmillan, London.
- Freathy, P. and O'Connell, F. (1999). A Typology of European Airport Retailing. *The Service Industries Journal*. 19:3, 119-134.
- Freathy, P. and O'Connell, F. (2000a). Market segmentation in the European airport sector. *Marketing Intelligence & Planning*. 18/3 [2000], 102-111.
- Freathy, P. and O'Connell, F. (2000b). Strategic Reactions to the Abolition of Duty Free: Examples from the European Airport Sector. *European Management Journal*, Vol. 18, No. 6, pp. 638–645, 2000.
- Freathy, Paul. (2004). The commercialisation of European airports: successful strategies in a decade of turbulence? *Journal of Air Transport Management*, Vol. 10, 191–197.
- Geuens, M., Vantomme, D. and Brengman, M. (2004). Developing a typology of airport shoppers. *Tourism Management*, Vol. 25 (2004) 615–622.
- Glazer, R. (1991). Marketing in an Information-Intensive Environment: Strategic Implications of Knowledge as an Asset. *Journal of Marketing*; Oct 1991; 55, 4.
- Gordon, J.R. and Gordon, St. R. (2000). Structuring The Interaction Between IT And Business Units. *Information Systems Management*, Winter 2000.
- Gosling, G.D. (1990). Design of an expert system for aircraft gate assignment. *Transportation Research-A*, Vol. 24 (1), 59-69.
- Graham, A. (2008). How important are commercial revenues to today's airports? *Journal of Air Transport Management*. (in press, doi: 10.1016/j.jairtraman.2008.11.004)
- Gu, Y., Chung, C.A. (1999). Genetic algorithm approach to aircraft gate reassignment problem. *Journal of Transportation Engineering* 125, 384–389.
- Haghani, A. and Chen, M.C. (1998). Optimizing gate assignments at airport terminals. *Transportation Research-A*, Vol. 32 (6), 437-454.

- Hamzawi, S.G. (1986). Management and planning of airport gate capacity: A microcomputer-based gate assignment simulation model. *Transportation Planning and Technology*, Vol. 11, 189–202.
- Hassounah, M.I. and Steuart, G.N. (1993). Demand for Aircraft Gates. *Transportation Research Record*, Vol. 1423, 26–33.
- Henderson, J.C. and Venkatraman, N. (1993). Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, Vol. 38, No. 2&3, 1999, 472-484, (Reprint from Vol. 32, No. 1, 1993).
- Hildebrandt, L. (1988). Store Image and the Prediction of Performance in Retailing. *Journal of Business Research*, Vol. 17, pp. 91-100 (1988)
- Hill, L. (2002). Digital Airport. *Air Transport World*, September 2002, 63-66.
- Hsu, Ch.-I. and Chao, Ch.-Ch. (2005). Space allocation for commercial activities at international passenger terminals. *Transportation Research Part E*, Vol. 41 (2005) 29–51.
- Hu, Q. and Plant, R. T. (2001). An empirical study of the causal relationship between IT investment and firm performance. *Information Resource Management Journal*, Vol. 14(3), 15–26.
- Hu, Q. and Quan, J. (2005). Evaluating the impact of IT investments on productivity: a causal analysis at industry level. *International Journal of Information Management*, Vol. 25, 39–53.
- Hu, X.-B. and Di Paolo, E. (2007). An Efficient Genetic Algorithm with Uniform Crossover for the Multi-Objective Airport Gate Assignment Problem. *Conference Paper at 2007 IEEE Congress on Evolutionary Computation (CEC2007)*, 25-28 Sep 2007, Singapore.
- Jarach, David. (2001). The evolution of airport management practices: towards a multi-point, multi-service, marketing-driven firm. *Journal of Air Transport Management*, Vol. 7, 119-125.
- Kaplan, R.S. and Norton, D.P. (1992). The Balanced Scorecard: Measures That Drive Performance. *Harvard Business Review*, January-February 1992, 71-79.
- Kaplan, R.S. and Norton, D.P. (2000). Having Trouble with your Strategy? Then Map It. *Harvard Business Review*, September-October 2000, 167-176.
- Kaplan, R.S. and Norton, D.P. (2004). How Strategy Maps Frame an Organization's Objectives. *Finance Executive*, March-April 2004, 40-45.
- Kazda, A. and Caves, R.E. (2000). *Airport Design and Operation*. Elsevier Science Ltd. Oxford, United Kingdom.
- Kelemen, Z. (2005). Resource Management System – The first step to the airport information system integration. *PERIODICA POLYTECHNICA SER. TRANSP. ENG*, Vol. 33, NO. 1–2, PP. 15–24 (2005).
- Kerkloh, M. (2007). Munich Airport's Terminal 2: A successful airport-airline collaboration. *AIRPORT MANAGEMENT*, Vol. 1, NO. 4, 330–337 JULY–SEPTEMBER 2007.
- Kim, H. and Shin, J. (2001). A contextual investigation of the operation and management of airport concessions. *Tourism Management*, Vol. 23, 149–155.
- Klann, D. (2001). Impact of e-commerce on an airline's inventory systems with possibilities to monitor and reduce fraud. *Unpublished M.Sc. Thesis - Cranfield University, School of Mechanical Engineering*, 2001.
- Kohli, R., and Devaraj, S. (2003). Measuring information technology payoff: a meta analysis of structural variables in firm-level empirical research. *Information Systems Research*, 14(2), 127–145.
- Koreimann, D.S. (1995). *Grundlagen der Software-Entwicklung. 2., durchges. Aufl.* Oldenbourg, München.

- Lam, S.H., Cao, J.-M. and Fan, H. (2002). Development of an intelligent agent for airport gate assignment. *Journal of Air Transportation*, Vol. 7, No. 2 – 2002.
- Lam, W.H.K., Tam, M.L., Wong, S.C. and Wirasinghe, S.C. (2003). Wayfinding in the passenger terminal of Hong Kong International Airport. *Journal of Air Transport Management*, Vol. 9, 73-81.
- Lim, A., Rodrigues, B. and Zhu, Y. (2005). Airport Gate Scheduling with Time Windows. *Artificial Intelligence Review* (2005) 24:5–31.
- Loveman, G. W. (1994). An assessment of the productivity impact on information technologies. In T. J. Allen, & M. S. Morton, *Information technology and the corporation of the 1990s: Research studies*. Cambridge: MIT Press.
- Luftman, J. (2003). Assessing IT/Business Alignment. *Information Strategy: The Executive's Journal*. Fall 2003, 7-14.
- Luftman, J., Papp, R. and Brier, T. (1999). Enablers and Inhibitors of Business-IT Alignment. *Communications of the Association for Information Systems*, Vol. 1, Article 1, March 1999.
- Luftman, J.N., Lewis, P.R. and Oldach, S.H. (1993). Transforming the enterprise: The alignment of business and information technology strategies. *IBM Systems Journal*. Armonk: 1993, Vol. 32, No. 1; 198-221.
- Mack, R. (2002). How to Tell If a Strategic Statement Is 'Strategic'. *Research Note*, TU-18-0143. Gartner Research, Stamford, Connecticut.
- Mack, R. (2003a). Management Update: An Alternative Approach to Creating an IT Strategy. *Research Note*, IGG-01012003-04. Gartner Research, Stamford, Connecticut.
- Mack, R. (2003b). Real IT Strategies: Steps 1 to 4 — Laying a Foundation. *Research Note*, R-21-4074. Gartner Research, Stamford, Connecticut.
- Mack, R. and Frey, N. (2002). Six Building Blocks for Creating Real IT Strategies. *Research Note*, R-17-3607. Gartner Research, Stamford, Connecticut.
- Melchert, F. and Winter, R. (2004). The Enabling Role of Information Technology for Business Performance Management. *DSS2004 Conference Proceedings*. Institute of Information Management, University of St. Gallen, St. Gallen, Switzerland.
- Montealegre, R. (2000). De-Escalating Information Technology Projects: Lessons From The Denver International Airport. *MIS Quarterly*, Vol. 24, No. 3, 417-447.
- Murty, K.G., Wan, Y.W., Yu, V.F., Dann, J. and Lee, R. (2008). Developing a DSS for Allocating Gates to Flights At An International Airport. *Unpublished*. March 2008.
- Neufville, de R. (1994). The baggage system at Denver: prospects and lessons. *Journal of Air Transport Management*, Vol. 1, 229-236.
- Obata, T. (1979) The Quadratic Assignment Problem: Evaluation of Exact and Heuristic Algorithms. *Tech. Report TRS-7901*, Rensselaer Polytechnic Institute, Troy, New York.
- Omar, O. (2001). Airport Retailing: Examining Airline Passengers' Impulsive Shopping Behaviour. *Journal of Euromarketing*, Vol. 11(1) 2001.
- Park, Y. (1999). A methodology for establishing operational standards of airport passenger terminals. *Journal of Air Transport Management*, Vol. 5, 73-80.
- Park, Y. (2003). An analysis for the competitive strength of Asian major airports. *Journal of Air Transport Management*, Vol. 9, 353-360.
- Peak, D., Guynesa, C. St., Kroon, V. (2005). Information technology Alignment Planning—a case study. *Information & Management*, Vol. 42 (2005), 619–633.
- Peppard, J. (1998). IS/IT Management in the Global Enterprise: A Framework for Strategic

- Alignment. *Cranfield School of Management Working Papers*, SWP 9/98.
- Peppard, J. and Ward, J. (2004). Beyond strategic information systems: towards an IS capability. *Journal of Strategic Information Systems*, Vol. 13, 167–194.
- Peppard, J., Edwards, Chr. and Lambert, R. (1999). A Governance Framework for Information Management in the Global Enterprise. *Paper for BIT World'99 Conference*, Cape Town, South Africa.
- Pitt, M., Wai, F.K. and Teck, P.C. (2002). Technology selection in airport passenger and baggage systems. *Facilities*, Vol. 20, No. 10, 314-326.
- Pollalis, Y.A. (2003). Patterns of co-alignment in information-intensive organizations: business performance through integration strategies. *International Journal of Information Management*, Vol. 23, 469–492.
- Porter, M.E. (1980). *Competitive Strategy: Techniques for Analyzing Industries and Competitors*. Free Press, New York.
- Porter, M.E. (1998). *On Competition*. Harvard Business Review Book, Cambridge.
- Porter, M.E. and Millar, V.E. (1985). How Information Gives You Competitive Advantage. *Harvard Business Review*, JULY-AUGUST 1985, 2-13.
- Rau, K.G. (2004). The CIO Dashboard Performance Management Program: Measuring and Managing the Value of IT. *Information Strategy: The Executive's Journal*, Winter 2004, 6-17.
- Reinheimer, S. (1998). Marktorientierte elektronische Koordination zwischenbetrieblicher Geschäftsprozesse in der Luftfracht. *dissertation.de, Verlag im Internet*, ISBN 3-933342-11-2. Berlin, Germany.
- Ross, J.W. and Beath, C.M. (2002). Beyond the Business Case: New Approaches to IT Investment. *MIT Sloan Management Review*, 51-59.
- Rowley, J. and Slack, F. (1999). The retail experience in airport departure lounges: reaching for timelessness and placelessness. *International Marketing Review*, Vol. 16 No. 4/5, 1999, pp. 363-375.
- Seamster, Th.L. and Kanki, B.G. (2002). *Aviation Information Management - From Documents to Data*. Ashgate Publishing Limited. Aldershot, England.
- Shifrin, C.A. (1988). Gate assignment expert system reduces delays at United's hubs. *Aviation Week & Space Technology*. January 5, 148-149 (1988).
- Srihari, K. and Muthukrishnan, R. (1991). An expert system methodology for an aircraft-gate assignment. *Computers & Industrial Engineering*, Vol. 21(1-4), 101-105.
- Su, Y.Y. and Srihari, K. (1993). A knowledge based aircraft-gate assignment advisor. *Computers and Industrial Engineering*, Vol. 25, 123–126.
- Tosic, V. (1992). A review of airport passenger terminal operations analysis and modeling. *Transportation Research-A*, Vol. 26 (1), 3-26.
- Vanderstraetan, G. and Bergeron, M. (1988). Automatic assignment of aircraft to gates at a terminal. *Computers and Industrial Engineering*, Vol.14, 15–25.
- Wiese, P. (2003a). A CEO Perspective on Strategic Use of IT. Aerospace and Defense - 2003 Information Technology Survey. *Experience, Results*. Computer Sciences Corporation (CSC), El Segundo, California, USA.
- Wiese, P. (2003b). Summary: Aerospace and Defense - 2003 Information Technology Survey. *Experience, Results*. Computer Sciences Corporation (CSC), El Segundo, California, USA.
- Wu, Ch.-L. (2005). Inherent delays and operational reliability of airline schedules. *Journal of Air Transport Management* 11 (2005) 273–282.

- 
- Wu, Ch.-L. and Caves, R.E. (2000). Aircraft operational costs and turnaround efficiency at airports. *Journal of Air Transport Management*, Vol. 6, 201-208.
- Yan, S. and Chang, C.M. (1998). A network model for gate assignment. *Journal of Advanced Transportation*, Vol. 32, 176-189.
- Yan, S. and Huo, C.M. (2001). Optimization of multiple objective gate assignments. *Transportation Research A*, Vol. 35, 413-432.
- Yan, S. and Tang, C.H. (2007). A heuristic approach for airport gate assignments for stochastic flight delays. *European Journal of Operational Research*, Vol. 180 (2007) 547-567.
- Yan, S., Chang, K.Ch. and Tang, Ch.-H. (2005). Minimizing inconsistencies in airport common-use checking counter assignments with a variable number of counters. *Journal of Air Transport Management*, Vol. 11 (2005) 107-116.
- Yan, S., Shieh, C.W., Chen, M. (2002). A simulation framework for evaluating airport gate assignments. *Transportation Research A*, Vol. 36, 885-898.
- Yan, S., Tang, C.H. and Chen, M. (2004). A model and a solution algorithm for airport common use check-in counter assignments. *Transportation Research Part A*, Vol. 38 (2004) 101-125.
- Zachman, J.A. (1987). A Framework for Information Systems Architecture. *IBM Systems Journal*, Vol. 26, no. 3, 1987. *IBM Publication G321-5298*.
- Zhang, S.X., Cesarone, J. and Miller, F.G. (1994). Comparative study of an aircraft assignment problem at a large International airport. *Journal of Industrial Engineering*, (13): 203-212.

## 7 Bibliography

- Aaker, David A. (2001). *Strategic Market Management*. 6<sup>th</sup> Ed., John Wiley & Sons, Inc. New York, USA.
- Abdelghanya, A., Abdelghanyb, K. and Narasimhanc, R. (2006). Scheduling baggage-handling facilities in congested airports. *Journal of Air Transport Management*, Vol. 12 (2006) 76–81.
- Abeyratne, Ruwantissa I.R. (2001). Revenue and investment management of privatized airports and air navigation services - a regulatory perspective. *Journal of Air Transport Management*, Vol. 7, 217-230.
- Airport Council International, 2007. *ACI Airport Economics Survey 2006*. ACI, Geneva.
- Alexopoulos, E. and Theodoulidis, B. (2003). The generic information business model. *International Journal of Information Management*, Vol. 23, 323–336.
- Anderson, R. (2003). MSBs Must Prioritize to Reduce Application Portfolio Costs. *Research Note*, COM-20-3434. Gartner Research, Stamford, Connecticut.
- Angeles, R., Corritore, C.L., Basu, S.C. and Nath R. (2001). Success factors for domestic and international electronic data interchange (EDI) implementation for US firms. *International Journal of Information Management*, Vol. 21, 329–347.
- Antoniou, P.H., Ansoff, H.I. (2004). Strategic Management of Technology. *Technology Analysis & Strategic Management*, Vol. 16, No. 2, 275–291, June 2004
- Appel, W. (2003). Enterprise Architecture: An In-Depth Study. *Paper of EPAS, Meta-Group*.
- Bailey, K. and Francis, K. (2008). Managing information flows for improved value chain performance. *International Journal of Production Economics*, Vol. 111 (2008) 2–12.
- Banks, T. (2008). Design work booms with rise in airport retail spend. *Design Week*. 28.08.2008.
- Barnes, Stuart J. (2002). The mobile commerce value chain: analysis and future developments. *International Journal of Information Management*, Vol. 22, 91–108.
- Baum, C. (2003). Designing Government Enterprise Architecture. *Research Note*, TU-20-5535. Gartner Research, Stamford, Connecticut.
- Bergey, J.K., Northrop, L.M., Smith, D.B. (1999). Enterprise Framework for the Disciplined Evolution of Legacy Systems. *Information Strategy - The Executive's Journal*, Vol. 16, No. 1, 15-35.
- Braganza, A. (2004). Rethinking the data–information–knowledge hierarchy: towards a case-based model. *International Journal of Information Management*, Vol. 24, 347–356.
- Browning, J. (2002). Three Pillars to SMB IT Value: Strategy, Architecture and Organization. *Research Note*, LE-17-1208. Gartner Research, Stamford, Connecticut.
- Buchanan, R. (Unknown). Enterprise Architecture and Strategic Planning Synergies. *EPAS, META Group*.
- Buytendijk, F. (2004a). Every Scorecard needs a strategy map. *Research Note*, COM-22-6893. Gartner Research, Stamford, Connecticut.
- Buytendijk, F. (2004b). High-Performance Organizations Need A Values-Based Scorecard. *Research Note*, DF-23-5153. Gartner Research, Stamford, Connecticut.
- Capri, S., Ignaccolo, M. (2004). Genetic algorithms for solving the aircraft-sequencing problem: the introduction of departures into the dynamic model. *Journal of Air Transport Management*, Vol. 10, 345–351.

- Centre for Airport Studies. (2001). *Airport Retail Study 2000/2001*. Arthur Anderson Services Practice, Sydney.
- Dallas, S. (2002). CIO Update: IT Governance Rules to Boost IS Organization and Business Unit Credibility. *Research Note*, IGG-12042002-03. Gartner Research, Stamford, Connecticut.
- Daugherty, P.J., Richey, G.R., Genchev, St.E. and Chen, H. (2005). Reverse logistics: superior performance through focused resource commitments to information technology. *Transportation Research Part E*, Vol. 41, 77–92.
- Davies, G. (1995). Bringing stores to shoppers – not shoppers to stores. *International Journal of Retail & Distribution Management*, Volume 23, Number, 1, 1995, 18–23.
- Dehninga, B. and Stratopoulos, Th. (2003). Determinants of a sustainable competitive advantage due to an IT-enabled strategy. *Journal of Strategic Information Systems*, Vol. 12, 7–28.
- DeJarnett, L. (2004). Notes From The Editor: The Great Debate: Does IT Really Matter?. *Information Strategy: The Executive's Journal*, Spring 2004, 3-5.
- Doganis, R. (2001). *The airline business in the 21st century*. Routledge, London.
- Drakos, N. and Casonato, R. (2002). Maintaining Flexibility Through Architectural Choices. *Research Note*, COM-16-8855. Gartner Research, Stamford, Connecticut.
- Drobik, A. (2002). Enterprise Architecture: The Business Issues and Drivers. *Research Note*, AV-17-3971. Gartner Research, Stamford, Connecticut.
- Duggan, J. and Light, M. (2002). *AD Portfolio Management: Tolerate, Integrate or Eliminate*. Research Note, COM-15-2917. Gartner Research, Stamford, Connecticut.
- Dürr, E. and Giannopoulos, G.A. (2003). SITS: a system for uniform intermodal freight transport information exchange. *International Journal of Transport Management*, Vol. 1, 175–186.
- Eastwood, G. (2008). Building the Responsive Enterprise – The evolution of BI, CRM and integrated information management. *Business Insights Technology Reports*, November 2008.
- Feiman, J. (2003). Selecting Alternatives in IT: A Decision-Making Model. *Research Note*, DF-18-7438. Gartner Research, Stamford, Connecticut.
- Feld, Ch.S. and Stoddard, D.B. (2004). Getting IT Right. *Harvard Business Review*, February 2004, 72-79.
- Feldman, J. (2002). First Class IT-Service. *Network Computing*, 17 April 2002, 44-49.
- Feldman, J.M. (1999). Controlling the Airport Data Grid. *Air Transport World*, 6/99, 35-42.
- Fenn, J. and Linden, A. (2003). Take the Initiative When Planning Your IT Strategy. *Research Note*, LE-20-7215. Gartner Research, Stamford, Connecticut.
- Fiala, P. (2005). Information sharing in supply chains. *Omega*, Vol. 33, 419–423.
- Fine, Ch.H. and Porteus, E.L. (1989). Dynamic Process Improvement. *Operations Research*, Vol. 37, No. 4, July-August 1989, 580-591.
- Francis, G., Humphreys, I. and Fry, J. (2002). The benchmarking of airport performance. *Journal of Air Transport Management*, Vol. 8, 239-247.
- Francis, G., Humphreys, I. and Fry, J. (2005). The nature and prevalence of the use of performance measurement techniques by airlines. *Journal of Air Transport Management*, Vol. 11, 207–217.
- Fuglseth, A.M. and Grønhaug, K. (2002). Theory-driven construction and analysis of cause maps. *International Journal of Information Management*, Vol. 22, 357–376.
- Fulton, R. (2003). Defining the Business Value of IT. *Research Note*, DF-18-3219. Gartner Research, Stamford, Connecticut.



- Gerrard, M. (2000). Minimizing IT Investment Risk Upfront. *Research Note*, TU-09-9047. Gartner Research, Stamford, Connecticut.
- Gerrard, M. (2001). Managing Risk When Making IT Investment Decisions. *Research Note*, SPA-14-3116. Gartner Research, Stamford, Connecticut.
- Gerrard, M. (2003). CIO Update: How to Make IT Centralization a Success. *Research Note*, IGG-01292003-03. Gartner Research, Stamford, Connecticut.
- Gillen, D. and Lall, A. (2002). The economics of the Internet, the new economy and opportunities for airports. *Journal of Air Transport Management*, Vol. 8, 49-62.
- Gomolski, B. (2003a). Management Update: Enterprises Should Assess How Their IT Spending Stacks Up. *Research Note*, IGG-08132003-01. Gartner Research, Stamford, Connecticut.
- Gomolski, B. (2003b). Managing the Finances of the IS Internal Service Company. *Research Note*, COM-19-4393. Gartner Research, Stamford, Connecticut.
- Gomolski, B. (2004). Ensure the Accuracy of Your IT Spending Analysis. *Research Note*, COM-22-6998. Gartner Research, Stamford, Connecticut.
- Gomolski, B. and Mack, R. (2002a). Financial Management Is Key to IS Organizations' Success. *Research Note*, K-18-9798. Gartner Research, Stamford, Connecticut.
- Gomolski, B. and Mack, R. (2002b). IT Budgeting to Be Co-opted by Just-in-Time Funding. *Research Note*, COM-18-7157. Gartner Research, Stamford, Connecticut.
- Goodwin, R. (2002). Transportation Industry IT Spending, 2000-2005 (Executive Summary). *Research Note*, ITSV-WW-EX-0131. Gartner Research, Stamford, Connecticut.
- Gordon, J.R.M., Lee, P.-M., Lucas Jr. and Henry C. (2005). A resource-based view of competitive advantage at the Port of Singapore. *Journal of Strategic Information Systems*, Vol. 14, 69–86.
- Gorry, A.G., Scott M. and Michael S. (1989). A Framework for Management Information Systems. *Sloan Management Review*, Spring 1989, 49-61.
- Gregor, S., Martin, M., Fernandez, W., Stern, St. and Vitale, M. (2006). The transformational dimension in the realization of business value from information technology. *Journal of Strategic Information Systems*, Vol. 15 (2006) 249–270.
- Gunasekaran, A., Love, P.E.D., Rahimi, F. and Miele, R. (2001). A model for investment justification in information technology projects. *International Journal of Information Management*, Vol. 21, 349–364.
- Hamzaee, R.G. and Vasigh, B. (2000). A simple revenue-cost perspective on US airport operations. *Journal of Air Transport Management*, Vol. 6, 61-64.
- Holloway, St. (1997). *Straight and Level: Practical airline economics*. Ashgate Publishing, Aldershot, UK, (Reprinted 2000, twice).
- Holly, A. (2000). Non-Aviation Revenues. *Airport Magazine*, May-June 2000, 14-15.
- Howard, S. and Anderson, R. (2002). Strategy, Architecture and Communications Best Practices. *Research Note*, COM-17-3003. Gartner Research, Stamford, Connecticut.
- Humphreys, I. and Francis, G. (2002). Performance measurement: a review of airports. *International Journal of Transport Management*, Vol. 1, 79–85.
- Ittner, Chr.D. and Larcker, D.F. (2003). Comint Up Short On Nonfinancial Performance Measurement. *Harvard Business Review*, November 2003, 88-95.
- James, G. (2002). Business Processes: A Compass for Architecture. *Research Note*, COM-16-8746. Gartner Research, Stamford, Connecticut.
- Jones, N. (2002). Why Are IT Investments So Often a Disappointment? *Research Note*, COM-16-5005. Gartner Research, Stamford, Connecticut.

- Kaipia, R. and Hartiala, H. (2006). Information-sharing in supply chains: five proposals on how to proceed. *The International Journal of Logistics Management*, Vol. 17 No. 3, 2006, pp. 377-393.
- Kemppainen, K., Nieminen J. and Vepsäläinen, A.P.J. (2007). Estimating the costs of airport congestion due to fast connections. *Journal of Air Transport Management*, Vol. 13 (2007) 169-174.
- Knox, M. (2003). Cost-Only Infrastructure Omits Future Revenue Potential. *Research Note*, SPA-20-0427. Gartner Research, Stamford, Connecticut.
- Kocharekar, R. (2004). An IT Architecture For Nimble Organizations: Managing Access From Cyberspace. *Information Systems Management*, Spring 2004, 22-30.
- Kohl, N., Larsen, A., Larsen, J. Ross, A. and Tiourine, S. (2007). Airline disruption management – perspectives, experiences and outlook. *Journal of Air Transport Management*, Vol. 13 (2007) 149-162.
- Kreizman, G. (2003). IT Governance in Transition. *Research Note*, AV-19-0737. Gartner Research, Stamford, Connecticut.
- Kulatilaka, N. and Venkatraman, N. (2001). Strategic Options in the Digital Era. *Business Strategy Review*, Vol. 12, No. 4, 7-15.
- Lapkin, A. and Rosser, B. (2003). Architecture Is Not About Technology. *Research Note*, TG-20-2290. Gartner Research, Stamford, Connecticut.
- Lim, D. and Palvia, P.C. (2001). EDI in strategic supply chain: impact on customer service. *International Journal of Information Management*, Vol. 21, 193-211.
- Loebbecke, C. and Powell, P. (1998). Competitive Advantage from IT in Logistics: The Integrated Transport Tracking System. *International Journal of Information Management*, Vol. 18 (1), 17-27.
- Machuca, J.A.D. and Barajas, R.P. (2004). The impact of electronic data interchange on reducing bullwhip effect and supply chain inventory costs. *Transportation Research Part E*, Vol. 40 (2004) 209-228.
- Mahoney, J. (2003). Consolidating IT: How to Score the Benefits and Barriers. *Research Note*, TG-19-4710. Gartner Research, Stamford, Connecticut.
- Mangoubi, R.S. and Mathaisel, D.F.X. (1985). Optimizing gate assignment at airport terminals. *Transportation Science*, Vol. 19 (2), 173-188.
- Matlus, R. (2004). Developing SLAs to Demonstrate the Business Value of IT. *Research Note*, COM-21-7511. Gartner Research, Stamford, Connecticut.
- Miller, D.J. (2006). Technological diversity, related diversification, and firm performance. *Strategic Management Journal*, Vol. 27: 601-619 (2006).
- Moodie, M. (2007). Analysing airport commercial revenues. In: Airport Council International. *ACI Airport Economics Survey 2006*. 20-25.
- Murphy, T. (2002). Focusing Purely on Finance is Bad for Your Financials. *Gartner G2 Report*. September 2002. Gartner Research, Stamford, Connecticut.
- Murray, J.P. (2004). Judging IT Department Performance. *Information Systems Management*, Spring 2004, 72-77.
- Newman, S. and Lloyd Jones, T. (1999). *Airport and Travel Terminal Retailing: Strategies, Trends and Market Dynamics*. Ravenfox Publishing, London.
- Page, S.N. and Rivera, Ivan. (2003). Relocating, Reallocating and Rethinking Concession Space. *Airport Magazine*, January-February 2003, 14-15.

- Palmer, J.W. and Griffith, D.A. (1998). Information intensity: A paradigm for understanding web site design. *Journal of Marketing Theory and Practice*. Summer 1998; 6, 3; p. 38.
- Panarella, A. (2002a). CEO and CIO Update: Case Studies Showing How IT Contributes to Business Success. *Research Note*, IGG-07312002-03. Gartner Research, Stamford, Connecticut.
- Panarella, A. (2002b). How IT Contributes to Business Success: Case Studies. *Research Note*, COM-16-6873. Gartner Research, Stamford, Connecticut.
- Pantazi, M.-A.A. and Georgopoulos, N.B. (2006). Investigating the impact of business-process-competent information systems (ISs) on business performance. *Managing Service Quality*, Vol.16 No. 4, pp. 421-434.
- Peacock, E. and Tanniru, M. (2005). Activity-based justification of IT investments. *Information & Management*, Vol. 42, 415–424.
- Prakash, A.C. (1998). Leveraging the Potential of Strategic Systems. *Information Systems Management*, Winter 1998.
- Raval, V. (2003). Productivity and Information Technology. *Information Strategy: The Executive's Journal*, Fall 2003, 37-40.
- Rivard, S., Raymond, L. and Verreault, D. (2006). Resource-based view and competitive strategy: An integrated model of the contribution of information technology to firm performance. *Journal of Strategic Information Systems*, Vol. 15 (2006) 29–50.
- Roberts, J. (2002). The Elusive Business Value of IT. *Research Note*, AV-17-2862. Gartner Research, Stamford, Connecticut.
- Roberts, J. (2003a). A Holistic Framework for Business Excellence. *Research Note*, TU-19-5797. Gartner Research, Stamford, Connecticut.
- Roberts, J. (2003b). The Business Value of IT Vendors. *Research Note*, DF-18-9870. Gartner Research, Stamford, Connecticut.
- Roberts, J. (2003c). The Business Value of the IT Application Portfolio. *Research Note*, DF-18-9665. Gartner Research, Stamford, Connecticut.
- Rosser, B. (2001). The Gartner Portfolio Management Tool for IT Investment. *Research Note*, TU-14-0675. Gartner Research, Stamford, Connecticut.
- Rosser, B. (2002a). A Practical Format for IT Architecture Guidelines. *Research Note*, TU-14-4319. Gartner Research, Stamford, Connecticut.
- Rosser, B. (2002b). Developing an Outline for Strategic IS Plans. *Research Note*, TU-15-9367. Gartner Research, Stamford, Connecticut.
- Rosser, B. (2002c). Mapping Architectural Styles to the Enterprise Framework. *Research Note*, COM-16-9013. Gartner Research, Stamford, Connecticut.
- Rosser, B. (2002d). What Is an Architectural Style? *Research Note*, COM-17-7016. Gartner Research, Stamford, Connecticut.
- Rosser, B. (2003a). Business Drivers as Guides to Architecture Choices. *Research Note*, AV-20-4094. Gartner Research, Stamford, Connecticut.
- Rosser, B. (2003b). Business Process Styles Can Benefit From Architecture. *Research Note*, TG-20-1350. Gartner Research, Stamford, Connecticut.
- Rosser, B. (2003c). Justifying Investments in Enterprise IT Architecture. *Research Note*, COM-19-4452. Gartner Research, Stamford, Connecticut.

- Salaün, Y. and Flores, K. (2001). Information quality: meeting the needs of the consumer. *International Journal of Information Management*, Vol. 21, 21-37.
- Salmela, H. and Spil, T.A.M. (2002). Dynamic and emergent information systems strategy formulation and implementation. *International Journal of Information Management*, Vol. 22, 441-460.
- Salwen, J. (2002). Enterprise Operations Management - Business Process Framework. *Auerbach Publications*, 8/02, 41-01-36.
- Santos, B.D. and Sussman, L. (2000). Improving the return on IT investment: the productivity paradox. *International Journal of Information Management*, Vol. 20, 429-440.
- Schulman, J. (2003). Ground Your Architecture in the Needs of Your Business. *Research Note*, LE-20-4594. Gartner Research, Stamford, Connecticut.
- Schulte, R. and Natis, Y. (2003). Event-Driven Architecture Complements SOA. Decision Framework. *Research Note*. DF-20-1154. Gartner Research, Stamford, Connecticut.
- Sechrest, L. (2002). Aligning IT and Business Objectives: The Role of Surveys. *Research Note*, COM-16-9154. Gartner Research, Stamford, Connecticut.
- Seneviratne, P.N. and Martel, N. (1991). Variables influencing performance of air terminal buildings. *Transportation Planning and Technology*. Vol. 16, 1177-1179.
- Shaw, St. (1999). *Airline Marketing and Management*. 4<sup>th</sup> Ed., Ashgate Publishing, Aldershot, UK. (Reprinted 2000).
- Silva, L., Figueroa, E.B. and Gonzalez-Reinhart, J. (2007). Interpreting IS alignment: A multiple case study in professional organizations. Information and Organization. *Information and Organization*, Vol. 17 (2007) 232-265.
- Sinur, J. (2002). Business Models: The Architecture That Pays for Itself. *Research Note*, COM-17-0438. Gartner Research, Stamford, Connecticut.
- Sowa, J.F. and Zachman, J.A. (1992). Extending and Formalizing the Framework for Information Systems Architecture. *IBM Systems Journal*, Vol. 31, no. 3, 1992. IBM Publication G321-5488.
- Spanos, Y.E., Lioukas, S. (2001). An examination into the causal logic of rent generation: contrasting Porter's competitive strategy framework and the resource-based perspective. *Strategic Management Journal*, Vol. 22: 907-934 (2001).
- Stamatopoulos, M.A., Zografos, K.G. and Odoni, A.R. (2004). A decision support system for airport strategic planning. *Transportation Research, Part C*, Vol. 12, 91-117.
- Stelter, D., Fechtel, A., Desai, P., Deimler, M., Koehler, M. and Sutherland, G. (2004). Airports - Dawn of a New Era. *Report. The Boston Consulting Group GmbH*, Munich, Germany.
- Stross, A. and Page, S.N. (2001). The Lure of Concession Revenues: How Do U.S. Airports Really Compare? *Airport Magazine*, September-October 2001, 20-22.
- Taller, M. (2002). A Guide For The Canadian Retail Travel Services Industry – New Strategies and Business Models. *Report. Trellis Consulting, the Trellis Group, Inc.*
- Talon, P.P. (2007). Does IT pay to focus? An analysis of IT business value under single and multi-focused business strategies. *Journal of Strategic Information Systems*, Vol. 16 (2007) 278-300.
- Tarafdar, M. and Gordon, St.R. (2007). Understanding the influence of information systems competencies on process innovation: A resource-based view. *Journal of Strategic Information Systems*, Vol. 16 (2007) 353-392.
- Thornton, C. (1999). Are We Aligned Yet?. *Auerbach Publications: IT Performance Improvement*, Volume 1, No. 12, 1+4-8.

- Tiboldi, T. (2008). Marketing strategy and airport revenue at FlyBalaton Airport. *AIRPORT MANAGEMENT*, Vol. 2, NO. 2, 153–157 JANUARY–MARCH 2008.
- Tingling, P. and Parent, M. (2004). An exploration of enterprise technology selection and evaluation. *Journal of Strategic Information Systems*, Vol. 13, 329–354.
- Vail III, E.F. (2002). Causal Architecture: Bringing The Zachman Framework To Life. *Information Systems Management*, Summer 2002, 8-19.
- Vasigh, B. and Hamazae, R.G. (1998). A comparative analysis of econometric performance of US commercial airports. *Journal of Air Transport Management*, Vol. 4, 209-216.
- Vasigh, B. and Haririan, M. (2003). An Empirical Investigation of Financial and Operational Efficiency of Private Versus Public Airports. *Journal of Air Transportation*, Vol. 8, No. 1, 91-110.
- Wagner, Chr. (2004). Enterprise strategy management systems: current and next generation. *Journal of Strategic Information Systems*, Vol. 13, 105–128.
- Wainwright, D. and Waring, T. (2004). Three domains for implementing integrated information systems: redressing the balance between technology, strategic and organisational analysis. *International Journal of Information Management*, Vol. 24, 329–346.
- Wallace, M. (1995). Survey: Practical Applications of Constraint Programming. *Unpublished*. Imperial College, London. September 1995.
- Walsh, P. (1996). Finding key performance drivers: Some new tools. *TOTAL QUALITY MANAGEMENT*, Vol. 7, No. 5, 509-519.
- Wijnhoven, F. Spil, T. Stegwee, R. and Tjang, R.A.Fa. (2006). Post-merger IT integration strategies: An IT alignment perspective. *Journal of Strategic Information Systems*, Vol. 15 (2006) 5–28.
- Woodward, G. (2002). A departure for retailers. *Design Week*. 14 November 2002.
- Wybolt, N.J. (1999). Information Management: Strategy, Systems and Technology. The Workgroup Model for Enterprise Architectures. *Auerbach Publications*, 8/99, 3-01-65.
- Xu, J.F. and Bailey, G. (2001). “The airport gate assignment problem: Mathematical model and a tabu search algorithm,” *Proceedings of the 34th Hawaii International Conference on System Sciences*, IEEE, p.10.
- Zhang, A. and Zhang Y. (2001). Airport charges and cost recovery: the long-run view. *Journal of Air Transport Management*, Vol. 7, 75-78.
- Zhang, T. and Zhang, D. (2007). Agent-based simulation of consumer purchase decision-making and the decoy effect. *Journal of Business Research*, Vol. 60 (2007) 912–922.

## 8 APPENDIX A

### 8.1 Overview of functions and procedures



Figure 72: Screenshot of software's function/procedure list. Source: Author.

## **8.2 Source code of software**

The software has been written in the PowerBasic programming language. The programming editor allows for syntax colouring and very long lines. Due to the thesis' format restrictions many lines of source code will be broken into the next line.

The source code of some functions that are not deemed necessary here has not been included in the following printout.

## 8.2.1 Main File

```

-----
' MAIN PROGRAM: START
-----

FUNCTION PBMain PRIVATE AS LONG

'-----
'--- GLOBAL VARIABLES ---
'-----

'-----
' FILENAMES
'-----

'PATHS
'-----

GLOBAL PATH_APPLICATION AS STRING 'directory that contains the
application (.exe-file)
GLOBAL PATH_DATA AS STRING
'directory that contains all data files
GLOBAL PATH_HISTORY AS STRING
'directory that contains files that have been backed up (.txt -> datetimestamp.txt)
GLOBAL PATH_SCENARIOS AS STRING 'directory that contains results of scenarios (opti runs):
(1) daily flight schedules (2) daily allocations

'FILES
'-----
GLOBAL FILE_SUMMER AS STRING 'this is
the working/current version of the summer season
GLOBAL FILE_WINTER AS STRING 'this is
the working/current version of the winter season
GLOBAL FILE_SS AS STRING
'Seasonal Flight Plan (Summer)
GLOBAL FILE_WS AS STRING
'Seasonal Flight Plan (Winter)

'--- ORG DATA FILES -----

GLOBAL FILE_ORG AS STRING
'PreOrgFile (fixed field length)

'--- ERROR FILES -----

GLOBAL FILE_ERR AS STRING
'Error Queue for QS reasons
GLOBAL FILE_ERR2 AS STRING
'ErrQueue with airport codes that are not in reference list (FILE_AIRPORTS)
GLOBAL FILE_ERR_03 AS STRING
'ErrFile (flights with no ATD)
GLOBAL FILE_ERR_04 AS STRING
'ErrFile (flights with no actual pax)
GLOBAL FILE_ERR_05 AS STRING
'ErrFile (flights with no flight pax DF factor)
GLOBAL FILE_ERR_06 AS STRING 'ErrFile
(flights with a delay of 9999)
GLOBAL FILE_ERR_07 AS STRING
'ErrFile (flights of season flight plan with no match in OrgDataFile)
GLOBAL FILE_ERR_08 AS STRING
'ErrFile (flights with no DF factor)

'--- MISC FILES -----

GLOBAL FILE_WSC AS STRING 'File with A/C and WingSpanCodes
GLOBAL FILE_GROUNDTIME AS STRING 'File with A/C and average standard ground times
GLOBAL FILE_GATE AS STRING 'File with Gates used
GLOBAL FILE_GATE_INFRA AS STRING 'File with Description of Gates
GLOBAL FILE_FLIRTTORG AS STRING 'original FLIRT data (for verification and fallback)'
GLOBAL FILE_FLIRT AS STRING 'FLIRT data (for verification and fallback)
GLOBAL FILE_OAG AS STRING 'File with flight numbers and destination airports
GLOBAL FILE_DF_FLIGHT AS STRING 'File WITH Duty Free factors per flight (e.g. BA8733;88,0)
GLOBAL FILE_DF_COUNTRY AS STRING 'File WITH Duty Free factors per country (e.g.
GERMANY;73,5)
GLOBAL FILE_RETAILAREAFACTORS AS STRING 'File WITH factors FOR each retail area ( rel. Retail
factor, e.g. R2;3,4 )
GLOBAL FILE_RETAIL_AREA_DEF AS STRING 'File with the definition of the retail areas
GLOBAL FILE_REVPERPAX AS STRING 'File with Average (Retail) Revenue Per PAX
GLOBAL FILE_DF_RETAIL_FACTOR AS STRING 'Duty Free Retail Factor
GLOBAL FILE_GANTTVIEW AS STRING 'File used for import in EXCEL spreadsheet with conditional
formatting
GLOBAL FILE_TIMEINDEX AS STRING 'File with an index for times during a day (00:00, 00:05, .. 24:00)
GLOBAL FILE_HEATCAT AS STRING 'File that defines the category according to which the heat map is
generated
GLOBAL FILE_AIRPORTS AS STRING 'File with updated AP codes (Russian Federation,
Yugoslavia, ...)
GLOBAL FILE_AIRLINES AS STRING 'File with Airline Codes
GLOBAL FILE_PAX001 AS STRING 'PAX data file used for import into EXCEL for pax stats

```



---

```

GLOBAL FILE_OPTIPARAMETERS      AS STRING  'File with (GA) optimization parameters
GLOBAL FILE_AIRLINEALLIANCES    AS STRING  'File with members of airline alliances

'-----
' OTHER
'-----

GLOBAL DEBUG_COUNTER_1 AS DOUBLE
GLOBAL DEBUG_COUNTER_2 AS DOUBLE
GLOBAL DEBUG_COUNTER_3 AS DOUBLE
GLOBAL DEBUG_COUNTER_4 AS DOUBLE
GLOBAL DEBUG_COUNTER_5 AS DOUBLE

GLOBAL DEBUG_PRINT AS INTEGER

GLOBAL gHeatCat_B_From AS LONG
GLOBAL gHeatCat_B_To AS LONG

GLOBAL gNumberOfGates AS INTEGER
GLOBAL gNumberOfAircraftTypes AS INTEGER

GLOBAL RUNMODE AS STRING

GLOBAL gVarNumber_1_TimerFunc AS DOUBLE
GLOBAL gVarNumber_2_TimerFunc AS DOUBLE
GLOBAL gVarNumber_3_TimerFunc AS DOUBLE

GLOBAL gVarString_1_TimerFunc AS STRING
GLOBAL gVarString_2_TimerFunc AS STRING
GLOBAL gVarString_3_TimerFunc AS STRING

GLOBAL gTimeCalcNumber_1 AS LONG
GLOBAL gTimeCalcNumber_2 AS LONG
GLOBAL gTimeCalcNumber_3 AS LONG
GLOBAL gTimeCalcNumber_4 AS LONG

'--- FOR OPTIMIZATION RUNS ---

GLOBAL gDKGA_NumberOfRecords AS LONG

GLOBAL gDKGA_MinutesAtGateMINIMUM AS INTEGER  'in file FILE_OPTIPARAMETERS
GLOBAL gDKGA_MinutesAtGateMAXIMUM AS INTEGER  'in file FILE_OPTIPARAMETERS
GLOBAL gBufferIntervals AS INTEGER            'in file FILE_OPTIPARAMETERS

'-----

DIM lMenuItem      AS LOCAL LONG
DIM lResult        AS LOCAL LONG
DIM sMenu(1 TO %MAXMENUBUFFER) AS LOCAL STRING
LOCAL i AS INTEGER

RUNMODE = "CRANFIELD"

```

```

'-----
' SUBS / FUNCTIONS
'-----

FUNCTION TitleBarTime(lNotUsed&) AS LONG

    ConsoleTitle "Current Time: " + TIME$

    IF RIGHT$(TIME$,1) = "0" THEN
        ConsoleIcon %IDI_HAND
    ELSE
        ConsoleIcon %IDI_ASTERISK
    END IF

END FUNCTION

'-----

SUB SPLASHBOX(sText$)

    SplashBoxShow 1+%BOLD, _
        0, _
        %CONSOLE_CENTER, _
        %CONSOLE_CENTER, _
        sText$, _
        "" , _
        "" , _
        0, _
        %TRUE

    SLEEP 1000
    SplashBoxHide

END SUB

'-----

FUNCTION OpenFileDialog(BYVAL hWnd AS LONG, _           ' parent window
    BYVAL sCaption AS STRING, _                       ' caption
    BYREF sFileNames AS STRING, _                    ' filename
    BYVAL sInitialDir AS STRING, _                   ' start directory
    BYVAL sFilter AS STRING, _                      ' filename filter
    BYVAL sDefExtension AS STRING, _                ' default extension
    BYREF lFlags AS LONG) AS LONG                    ' flags

    DIM tOFN          AS LOCAL OPENFILENAME
    DIM sRetVal       AS LOCAL STRING

    sRetVal = STRING$(1024,0)
    MID$(sRetVal,1) = sFileNames

    REPLACE "|" WITH CHR$(0) IN sFilter

    IF LEN(sInitialDir) = 0 THEN
        sInitialDir = CURDIR$
    END IF

    tOFN.lStructSize      = SIZEOF(tOFN)
    tOFN.hWndOwner        = hWnd
    tOFN.lpstrFilter      = STRPTR(sFilter)
    tOFN.nFilterIndex     = 1
    tOFN.lpstrFile        = STRPTR(sRetVal)
    tOFN.nMaxFile        = LEN(sRetVal)
    tOFN.lpstrInitialDir  = STRPTR(sInitialDir)
    IF LEN(sCaption) THEN
        tOFN.lpstrTitle   = STRPTR(sCaption)
    END IF
    tOFN.Flags            = lFlags OR %OFN_ENABLEHOOK OR %OFN_EXPLORER
    tOFN.lpfnHook         = CODEPTR(OfnHook)
    tOFN.lpstrDefExt      = STRPTR(sDefExtension)

    'This function returns zero (0) if the user selects Cancel or Close.
    FUNCTION = GETOPENFILENAME(tOFN)

    sFileNames = sRetVal
    lFlags     = tOFN.Flags

END FUNCTION

'-----

'---- SYSTEM ADMIN FUNCTIONS ----
'-----

SUB LogEntry(BYVAL SenderFunction AS STRING, BYVAL LogText AS STRING)

    'writes information with a time stamp into a log file

```

```

DIM FileHandle AS INTEGER

FileHandle = FREEFILE

    OPEN PATH_APPLICATION+$FILE_ACTLOG FOR APPEND AS #FileHandle
    PRINT #FileHandle, DATE$ + ";" + TIME$ + ";" + SenderFunction + SPACE$(50-LEN(SenderFunction)) +
";" UCASE$(LogText)
    CLOSE #FileHandle
END SUB 'LogEntry()

'-----

SUB ShowWaitBox(BYVAL DurationSecs AS INTEGER)

    LOCAL hBmp AS LONG

    LOCAL h, w, hGW AS LONG

    h = 100
    w = 200

    GRAPHIC WINDOW "Processing...", LocOfCol(22), LocOfRow(5), w, h TO hGW

    GRAPHIC ATTACH hGW, 0&, REDRAW
    GRAPHIC COLOR RGB(0,0,0), RGB(255,255,255)
    GRAPHIC CLEAR

    GRAPHIC BITMAP LOAD PATH_APPLICATION+"WaitBox.bmp", 200, 100 TO hBmp
    GRAPHIC COPY hBmp, 0 TO (1, 1)
    GRAPHIC REDRAW

    SLEEP DurationSecs*1000

    GRAPHIC BITMAP END
    GRAPHIC WINDOW END

END SUB 'ShowWaitBox()

'-----

FUNCTION aeroCUBEInit() AS INTEGER

'for initialization at start of program
LOCAL FUNCTION_PART AS INTEGER
LOCAL FUNCTION_PART2 AS INTEGER
LOCAL lErrorOccurred AS LONG
LOCAL VariableCounter AS INTEGER
LOCAL FoundIndexPosition AS LONG
LOCAL VariableNotFoundMsg AS STRING
LOCAL FileNotFoundTest AS LONG

LOCAL INI_Line AS STRING
LOCAL OpsDataLine AS STRING

LOCAL LineCounter AS LONG

'-----

lErrorOccurred = %FALSE
FUNCTION_PART = 1
FUNCTION_PART2 = 1
VariableCounter = 0

    FUNCTION = %TRUE 'return %FALSE if you want the program to end.

ON ERROR GOTO ErrorTrap

cls
PRINT
PRINT "Reading of .ini-File and initialization of variables..."

    CALL LogEntry(FUNCNAME$, "-----")
    CALL LogEntry(FUNCNAME$, "START PROGRAM")
    CALL LogEntry(FUNCNAME$, "-----")

'frist count number of valid variable entries in in-file -----

OPEN PATH_APPLICATION+$FILE_INI FOR INPUT AS #1
WHILE NOT EOF(1)
    LINE INPUT #1, INI_Line
    IF LEFT$(INI_Line, 2) <> "//" AND LEN(TRIM$(INI_Line)) > 0 THEN
        INCR VariableCounter
    END IF
WEND 'EOF(1)

```

```

CLOSE #1

IF VariableCounter <> %NumberOfGLOBALsUsed THEN
    ERROR %WrongNumberOfGlobalVars
END IF

FUNCTION_PART = 2

'then define an array and read variables -----
DIM aINI_Variables(1 TO VariableCounter, 1 TO 2) AS STRING
OPEN PATH_APPLICATION+$FILE_INI FOR INPUT AS #1

VariableCounter = 0

WHILE NOT EOF(1)

    LINE INPUT #1, INI_Line

    IF LEFT$(INI_Line, 2) <> "/" AND LEN(TRIM$(INI_Line)) > 0 THEN
        INCR VariableCounter
        aINI_Variables(VariableCounter, 1) = UCASE$(TRIM$(PARSE$(INI_Line, ";", 1))) 'Name of
Variable
        aINI_Variables(VariableCounter, 2) = UCASE$(TRIM$(PARSE$(INI_Line, ";", 2))) 'Content
of Variable
    END IF

WEND 'EOF(1)

CLOSE #1

'-----
'check on the 39 variables and initialize them -----
'-----

'+++ FILENAMES DEFINITIONS +++
'-----

'-----
'PATHS
'-----

'--- PATH_APPLICATION ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="PATH_APPLICATION", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    PATH_APPLICATION = aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: PATH_APPLICATION = " + PATH_APPLICATION)
    FileNotFoundTest = GETATTR(aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "PATH_APPLICATION"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- PATH_DATA ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="PATH_DATA", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    PATH_DATA = aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: PATH_DATA = " + PATH_DATA)
    FileNotFoundTest = GETATTR(aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "PATH_DATA"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- PATH_HISTORY ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="PATH_HISTORY", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    PATH_HISTORY = aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: PATH_HISTORY = " + PATH_HISTORY)
    FileNotFoundTest = GETATTR(aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "PATH_HISTORY"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- PATH_SCENARIOS ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="PATH_SCENARIOS", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    PATH_SCENARIOS = aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: PATH_SCENARIOS = " + PATH_SCENARIOS)
    FileNotFoundTest = GETATTR(aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "PATH_SCENARIOS"
    ERROR %APP_FILE_NOT_FOUND

```

```

END IF

'-----
'FILES
'-----

FUNCTION_PART2 = 2

'--- FILE_SUMMER ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_SUMMER", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_SUMMER = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_SUMMER = " + FILE_SUMMER)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_SUMMER"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_WINTER ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_WINTER", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_WINTER = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_WINTER = " + FILE_WINTER)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_WINTER"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_SS ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_SS", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_SS = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_SS = " + FILE_SS)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_SS"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_WS ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_WS", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_WS = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_WS = " + FILE_WS)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_WS"
    ERROR %APP_FILE_NOT_FOUND
END IF

'-----
'--- ORG DATA FILES ---
'-----

'--- FILE_ORG ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_ORG", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_ORG = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_ORG = " + FILE_ORG)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_ORG"
    ERROR %APP_FILE_NOT_FOUND
END IF

'-----
'--- ERROR FILES ---
'-----

'--- FILE_ERR ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_ERR", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_ERR = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_ERR = " + FILE_ERR)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_ERR"
    ERROR %APP_FILE_NOT_FOUND
END IF

```

```

'--- FILE_ERR2 ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_ERR2", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_ERR2 = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_ERR2 = " + FILE_ERR2)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_ERR2"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_ERR_03 ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_ERR_03", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_ERR_03 = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_ERR_03 = " + FILE_ERR_03)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_ERR_03"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_ERR_04 ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_ERR_04", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_ERR_04 = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_ERR_04 = " + FILE_ERR_04)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_ERR_04"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_ERR_05 ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_ERR_05", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_ERR_05 = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_ERR_05 = " + FILE_ERR_05)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_ERR_05"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_ERR_06 ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_ERR_06", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_ERR_06 = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_ERR_06 = " + FILE_ERR_06)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_ERR_06"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_ERR_07 ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_ERR_07", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_ERR_07 = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_ERR_07 = " + FILE_ERR_07)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_ERR_07"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_ERR_08 ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_ERR_08", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_ERR_08 = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_ERR_08 = " + FILE_ERR_08)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_ERR_08"
    ERROR %APP_FILE_NOT_FOUND
END IF

'-----
'--- MISC FILES ---
'-----

```

```

'--- FILE_WSC ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_WSC", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_WSC = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_WSC = " + FILE_WSC)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))

'Count Number Of AircraftTypes and store in global variable (for multiple use in software)
    OPEN FILE_WSC FOR INPUT AS #1
LineCounter = 0
WHILE NOT EOF(1)
    LINE INPUT #1, OpsDataLine
    IF LEFT$(OpsDataLine, 2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN
        INCR LineCounter
    END IF
WEND 'eof(1)
CLOSE #1
gNumberOfAircraftTypes = LineCounter

ELSE
    VariableNotFoundMsg = "FILE_WSC"
    ERROR %APP_FILE_NOT_FOUND

END IF

'--- FILE_GROUNDTIME ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_GROUNDTIME", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_GROUNDTIME = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_GROUNDTIME = " + FILE_GROUNDTIME)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))

ELSE
    VariableNotFoundMsg = "FILE_GROUNDTIME"
    ERROR %APP_FILE_NOT_FOUND

END IF

'--- FILE_GATE ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_GATE", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_GATE = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_GATE = " + FILE_GATE)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))

ELSE
    VariableNotFoundMsg = "FILE_GATE"
    ERROR %APP_FILE_NOT_FOUND

END IF

'--- FILE_GATE_INFRA ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_GATE_INFRA", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_GATE_INFRA = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_GATE_INFRA = " + FILE_GATE_INFRA)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))

'Count Number Of Gates and store in global variable (for multiple use in software)
    OPEN FILE_GATE_INFRA FOR INPUT AS #1
LineCounter = 0
WHILE NOT EOF(1)
    LINE INPUT #1, OpsDataLine
    IF LEFT$(OpsDataLine, 2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN
        INCR LineCounter
    END IF
WEND 'eof(1)
CLOSE #1
gNumberOfGates = LineCounter

ELSE
    VariableNotFoundMsg = "FILE_GATE_INFRA"
    ERROR %APP_FILE_NOT_FOUND

END IF

'--- FILE_FLIRTOG ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_FLIRTOG", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_FLIRTOG = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_FLIRTOG = " + FILE_FLIRTOG)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))

ELSE
    VariableNotFoundMsg = "FILE_FLIRTOG"
    ERROR %APP_FILE_NOT_FOUND

END IF

'--- FILE_FLIRT ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_FLIRT", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_FLIRT = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)

```

```

        CALL LogEntry(FUNCNAME$, "Initialize: FILE_FLIRT = " + FILE_FLIRT)
        FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_FLIRT"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_OAG ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_OAG", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_OAG = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_OAG = " + FILE_OAG)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_OAG"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_DF_FLIGHT ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_DF_FLIGHT", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_DF_FLIGHT = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_DF_FLIGHT = " + FILE_DF_FLIGHT)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_DF_FLIGHT"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_DF_COUNTRY ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_DF_COUNTRY", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_DF_COUNTRY = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_DF_COUNTRY = " + FILE_DF_COUNTRY)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_DF_COUNTRY"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_RETAILAREAFACTORS ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_RETAILAREAFACTORS", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_RETAILAREAFACTORS = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_RETAILAREAFACTORS = " + FILE_RETAILAREAFACTORS)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_RETAILAREAFACTORS"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_RETAIL_AREA_DEF ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_RETAIL_AREA_DEF", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_RETAIL_AREA_DEF = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_RETAIL_AREA_DEF = " + FILE_RETAIL_AREA_DEF)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_RETAIL_AREA_DEF"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_REVPERPAX ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_REVPERPAX", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_REVPERPAX = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_REVPERPAX = " + FILE_REVPERPAX)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_REVPERPAX"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_DF_RETAIL_FACTOR ---

FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_DF_RETAIL_FACTOR", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_DF_RETAIL_FACTOR = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_DF_RETAIL_FACTOR = " + FILE_DF_RETAIL_FACTOR)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_DF_RETAIL_FACTOR"

```



```

        ERROR %APP_FILE_NOT_FOUND
    END IF

    '--- FILE_GANTTVIEW ---

    FoundIndexPosition = 0
    ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_GANTTVIEW", TO FoundIndexPosition
    IF FoundIndexPosition <> 0 THEN
        FILE_GANTTVIEW = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
        CALL LogEntry(FUNCNAME$, "Initialize: FILE_GANTTVIEW = " + FILE_GANTTVIEW)
        FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
    ELSE
        VariableNotFoundMsg = "FILE_GANTTVIEW"
        ERROR %APP_FILE_NOT_FOUND
    END IF

    '--- FILE_TIMEINDEX ---

    FoundIndexPosition = 0
    ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_TIMEINDEX", TO FoundIndexPosition
    IF FoundIndexPosition <> 0 THEN
        FILE_TIMEINDEX = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
        CALL LogEntry(FUNCNAME$, "Initialize: FILE_TIMEINDEX = " + FILE_TIMEINDEX)
        FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
    ELSE
        VariableNotFoundMsg = "FILE_TIMEINDEX"
        ERROR %APP_FILE_NOT_FOUND
    END IF

    '--- FILE_AIRPORTS ---

    FoundIndexPosition = 0
    ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_AIRPORTS", TO FoundIndexPosition
    IF FoundIndexPosition <> 0 THEN
        FILE_AIRPORTS = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
        CALL LogEntry(FUNCNAME$, "Initialize: FILE_AIRPORTS = " + FILE_AIRPORTS)
        FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
    ELSE
        VariableNotFoundMsg = "FILE_AIRPORTS"
        ERROR %APP_FILE_NOT_FOUND
    END IF

    '--- FILE_AIRLINES ---

    FoundIndexPosition = 0
    ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_AIRLINES", TO FoundIndexPosition
    IF FoundIndexPosition <> 0 THEN
        FILE_AIRLINES = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
        CALL LogEntry(FUNCNAME$, "Initialize: FILE_AIRLINES = " + FILE_AIRLINES)
        FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
    ELSE
        VariableNotFoundMsg = "FILE_AIRLINES"
        ERROR %APP_FILE_NOT_FOUND
    END IF

    '--- FILE_PAX001 ---

    FoundIndexPosition = 0
    ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_PAX001", TO FoundIndexPosition
    IF FoundIndexPosition <> 0 THEN
        FILE_PAX001 = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
        CALL LogEntry(FUNCNAME$, "Initialize: FILE_PAX001 = " + FILE_PAX001)
        FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
    ELSE
        VariableNotFoundMsg = "FILE_PAX001"
        ERROR %APP_FILE_NOT_FOUND
    END IF

    '--- FILE_HEATCAT ---

    FoundIndexPosition = 0
    ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_HEATCAT", TO FoundIndexPosition
    IF FoundIndexPosition <> 0 THEN
        FILE_HEATCAT = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
        CALL LogEntry(FUNCNAME$, "Initialize: FILE_HEATCAT = " + FILE_HEATCAT)
        FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))

        OPEN FILE_HEATCAT FOR INPUT AS #1
        WHILE NOT EOF(1)
            LINE INPUT #1, OpsDataLine
            IF LEFT$(OpsDataLine, 2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN
                gHeatCat_B_From = VAL(PARSE$(OpsDataLine, ";", 1))
                gHeatCat_B_To = VAL(PARSE$(OpsDataLine, ";", 2))
            END IF
        WEND 'EOF(1)
        CLOSE #1
    ELSE
        VariableNotFoundMsg = "FILE_HEATCAT"
        ERROR %APP_FILE_NOT_FOUND
    END IF

    '--- FILE_OPTIPARAMETERS ---

    FoundIndexPosition = 0
    ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_OPTIPARAMETERS", TO FoundIndexPosition

```

```

IF FoundIndexPosition <> 0 THEN
    FILE_OPTIPARAMETERS = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_OPTIPARAMETERS = " + FILE_OPTIPARAMETERS)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_OPTIPARAMETERS"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- FILE_AIRLINEALLIANCES ---
FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="FILE_AIRLINEALLIANCES", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    FILE_AIRLINEALLIANCES = PATH_DATA + aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: FILE_AIRLINEALLIANCES = " + FILE_AIRLINEALLIANCES)
    FileNotFoundTest = GETATTR(PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
ELSE
    VariableNotFoundMsg = "FILE_AIRLINEALLIANCES"
    ERROR %APP_FILE_NOT_FOUND
END IF

'--- RUNMODE ---
FoundIndexPosition = 0
ARRAY SCAN aINI_Variables(1,1) FOR VariableCounter, ="RUNMODE", TO FoundIndexPosition
IF FoundIndexPosition <> 0 THEN
    RUNMODE = aINI_Variables(FoundIndexPosition, 2)
    CALL LogEntry(FUNCNAME$, "Initialize: RUNMODE = " + RUNMODE)
ELSE
    VariableNotFoundMsg = "RUNMODE"
    ERROR %APP_FILE_NOT_FOUND
END IF

'check whether all files exist at the place according to ini-file -----
'-----
FF_WINMAIN_RESUME:
IF lErrorOccurred = %TRUE THEN
    ConsoleMessageBox "Terminating Program.",%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

        CALL LogEntry(FUNCNAME$, "-----")
        CALL LogEntry(FUNCNAME$, "EXIT PROGRAM: DURING INITIALIZATION")
        CALL LogEntry(FUNCNAME$, "-----")

    FUNCTION = %FALSE
END IF

cls

'-----
EXIT FUNCTION
'-----
ErrorTrap:

lErrorOccurred = %TRUE

SELECT CASE FUNCTION_PART

    CASE 1

        IF ERR = %WrongNumberOfGlobalVars THEN
            CALL LogEntry(FUNCNAME$, "initialize: wrong number of variables in .ini-file")
            ConsoleMessageBox "Number of variables: " + STR$(VariableCounter) + "\n" + _
                "(Expected: " + STR$(%NumberOfGLOBALSUsed) + ")\" + _
                "--> Check .ini-
file",%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
        ELSE
            ConsoleMessageBox "INI-file is
missing.",%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
        END IF

    CASE 2

        SELECT CASE ERR
            CASE %APP_FILE_NOT_FOUND
                ConsoleMessageBox "Variable not found in ini.file: " +
VariableNotFoundMsg,%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
            CASE %ERR_FILENOTFOUND, %ERR_PATHFILEACCESSERROR
                IF FUNCTION_PART2 = 1 THEN
                    ConsoleMessageBox "Not found on medium: " +
aINI_Variables(FoundIndexPosition, 2),%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
                    CALL LogEntry(FUNCNAME$, "initialize: not found on medium:
" + aINI_Variables(FoundIndexPosition, 2))
                ELSEIF FUNCTION_PART2 = 2 THEN
                    ConsoleMessageBox "Not found on medium: " + PATH_DATA +
aINI_Variables(FoundIndexPosition, 2),%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
                    CALL LogEntry(FUNCNAME$, "initialize: not found on medium:
" + PATH_DATA + aINI_Variables(FoundIndexPosition, 2))
                ELSE
                    ConsoleMessageBox "Not found on medium: " +
aINI_Variables(FoundIndexPosition, 2),%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

```

```

CALL LogEntry(FUNCNAME$, "initialize: not found on medium:
" + aINI_Variables(FoundIndexPosition, 2))
END IF

CASE ELSE
  ConsoleMessageBox "ERROR OCCURRED: " + STR$(ERR) + " " +
  ERROR$, %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
  CALL LogEntry(FUNCNAME$, "Else: " +
aINI_Variables(FoundIndexPosition, 2))
END SELECT

END SELECT

RESUME FF_WINMAIN_RESUME

'-----

END FUNCTION 'aeroCUBEInit()

'-----
'--- MAIN DATA FUNCTIONS ---
'-----

SUB LoadOpsData_Country(BYVAL FileName AS STRING)
'fills the destination country info into data file
'Lookup airport of destination according to flight number in OAG file
'if match, determine the country that belongs to the destination city code

  CALL LogEntry(FUNCNAME$, "START")

  '--- Declarations -----
  DIM OAG_RawData AS STRING

  DIM OAG_FlightNumber(%AmountOfFlights) AS STRING * 8
  DIM OAG_FlightDestCity(%AmountOfFlights) AS STRING * 3

  DIM lPosition AS LONG
  DIM APDataLine(%AmountOfAirports) AS STRING

  DIM AllRecs AS LONG

  DIM OpsDataLine AS STRING
  DIM FieldValue AS STRING
  DIM LineField(36) AS STRING
  DIM FieldCounter AS LONG
  DIM LineCounter AS LONG
  DIM i AS LONG 'general purpose loop counter
  DIM ArrayPointer AS LONG
  DIM lFoundAirport AS INTEGER

  DIM SecStart AS LONG
  DIM SecCurrent AS LONG

  LOCAL hWin AS DWORD ' To create and show a Graphic window on screen

  'Initialization -----
  OpsDataLine = ""
  FieldValue = ""
  FieldCounter = 0
  LineCounter = 0

  '-----
  'open OrgData csv-formatted file (season)
  OPEN FileName FOR INPUT AS #1

  'open OAG flight plan data for destination AP / country
  OPEN FILE_OAG FOR INPUT AS #3

  'open airport reference file with countries in it
  OPEN FILE_AIRPORTS FOR INPUT AS #2 'for third run with updated country info (Russia, Yugoslavia, ...)

  'open new target file
  OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #5

  'ErrQueue with airport codes that are not in reference list (AP_CODES.TXT)
  OPEN FILE_ERR2 FOR OUTPUT AS #6

  '-----

  GRAPHIC WINDOW "Processing...", 500, 300, 450, 130 TO hWin

```

```

GRAPHIC ATTACH hWin, 0
GRAPHIC SET FOCUS

SecStart = TIMER

'-----

'read airport reference data into array
LineCounter = 1
WHILE ISFALSE EOF(3) AND LineCounter < %AmountOfFlights
  LINE INPUT #3, OAG_RawData
  OAG_FlightNumber(LineCounter) = LEFT$(OAG_RawData, 8)
  OAG_FlightDestCity(LineCounter) = MID$(OAG_RawData, 17, 3)

  INCR LineCounter
WEND
CLOSE #3

FILESCAN #1, RECORDS TO AllRecs 'used for progress calculation

'read country reference data into array
LineCounter = 1
WHILE ISFALSE EOF(2) AND LineCounter < %AmountOfAirports

  LINE INPUT #2, APDataLine(LineCounter)

  'Structure of APDataLine
  'POS      Meaning
  '-----
  '01-03   = IATA 3-Letter-Code
  '09-12   = ICAO 4-Letter-Code
  '17-39   = City
  '40-68   = Country
  '69-EOL  = Airport Name

  INCR LineCounter

WEND 'EOF(2)
CLOSE #2

'read OrgData and enhance by country entry
LineCounter = 1

SecCurrent = TIMER
SecStart = SecCurrent

ArrayPointer = 1

WHILE ISFALSE EOF(1) 'AND LineCounter < 10 ' check if at end of file

  'show progress on screen each 10 seconds
  IF TIMER >= SecCurrent + 10 THEN
    SecCurrent = TIMER
    GRAPHIC SET POS (30, 30)
    GRAPHIC PRINT "Processing... " STR$(LineCounter) & " OF " & STR$(AllRecs) & " = " &
STR$(INT((LineCounter/AllRecs)*100)) & " % "
  END IF

  LINE INPUT #1, OpsDataLine
  'parse the OpsDataLine and fill the field variables

  FieldValue = ""
  FieldCounter = 1

  FOR i = 1 TO LEN(OpsDataLine)
    IF MID$(OpsDataLine, i,1) <> ";" THEN
      'still in same field
      FieldValue = FieldValue + MID$(OpsDataLine, i,1)
    ELSE
      'field changes, so fill content into LineFieldArray
      LineField(FieldCounter) = FieldValue
      INCR FieldCounter
      FieldValue = ""
    END IF
  NEXT i

  'Lookup airport of destination according to flight number
  'field 02 is flight number
  'ARRAY SCAN OAG_FlightNumber(), =LineField(02), TO lPosition

  lPosition = 0
  FOR i = 1 TO %AmountOfFlights
    IF OAG_FlightNumber(i) = LEFT$(LineField(03),8) THEN 'left$-function because of flight numbers like
"FUA00108F"
      lPosition = i
      i = %AmountOfFlights
    END IF
  NEXT i

  IF lPosition = 0 THEN

```

```

'no match found, so set country to "---"
LineField(27) = "---"

ELSE

'match, so determine the country that belongs to the destination city code
'Lookup destination country
'field 27 is destination country
LineField(27) = "..."
lFoundAirport = 0
FOR i = 1 TO %AmountOfAirports

'city codes match ?
IF LEFT$(APDataLine(i),3) = OAG_FlightDestCity(lPosition) THEN
'store determined country in according LineField
LineField(27) = RTRIM$(MID$(APDataLine(i),40,28))
i = %AmountOfAirports + 1
lFoundAirport = 1
END IF

NEXT i

IF lFoundAirport = 0 THEN
'if there has not been a match in the AP_CODES.TXT
PRINT #6, OAG_FlightDestCity(lPosition)
END IF

END IF

'Write enhanced line into new output file

OpsDataLine = ""
FOR i = 1 TO 36
OpsDataLine = OpsDataLine + TRIM$(LineField(i)) + ";"
NEXT i

PRINT #5, OpsDataLine

INCR LineCounter

WEND 'EOF(1)
GRAPHIC WINDOW END

CLOSE #1
CLOSE #5
CLOSE #6

CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))

CALL LogEntry(FUNCNAME$, "END")
END SUB 'LoadOpsData_Country()

'-----

SUB CalcDelayMinutes(BYVAL FileName AS STRING)

CALL LogEntry(FUNCNAME$, "START: " + FileName)

'calculate the difference between STD und ATD in minutes

DIM OpsDataLine AS STRING

DIM i AS LONG 'general purpose loop counter

'open season origin file
OPEN FileName FOR INPUT AS #1

'open new target file
OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #2

WHILE ISFALSE EOF(1)

LINE INPUT #1, OpsDataLine

OpsDataLine = StringUpdate(OpsDataLine, %FIELD_DelayMinutes, DelayMin(PARSE$(OpsdataLine, ";", %FIELD_STD),
PARSE$(OpsdataLine, ";", %FIELD_ATD) ) )

'write updated file
PRINT #2, OpsDataLine

WEND 'EOF(1)

CLOSE #1
CLOSE #2

CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))
CALL LogEntry(FUNCNAME$, "END: " + FileName)

END SUB 'CalcDelayMinutes()

```

```

-----
FUNCTION DelayMin(BYVAL STD AS STRING, BYVAL ATD AS STRING) AS STRING

  IF STD = "" OR ATD = "" THEN
    'missing value(s)
    DelayMin = "8888"
  ELSE
    IF LEFT$(STD,8) = LEFT$(ATD,8) THEN
      '--> same day
      'calc difference, e.g. 1630 and 1736
      DelayMin = TRIM$( STR$( ( (VAL(MID$(ATD,9,2)) * 60 ) + VAL(MID$(ATD,11,2)) ) - _
        ( (VAL(MID$(STD,9,2)) * 60 ) + VAL(MID$(STD,11,2)) ) ) - _
          ) )
    ELSE 'not same day
      IF LEFT$(ATD,8) = NextDay(LEFT$(STD,8)) THEN
        DelayMin = TRIM$( STR$( ( (23-VAL(MID$(STD,9,2))) * 60 ) + (60 - VAL(MID$(STD,11,2))) ) + _
          ( ( VAL(MID$(ATD,9,2)) * 60 ) + VAL(MID$(ATD,11,2)) ) ) - _
            ) )
      ELSE
        'if departure not even at next day then mark record with '9999'
        DelayMin = "9999"
      END IF
    END IF 'same day
  END IF
END FUNCTION 'DelayMin()
-----

```

```

-----
FUNCTION NextDay(BYVAL yyyyymmdd AS STRING) AS STRING
  'returns a string that represents the next day date of the input string
  LOCAL year AS INTEGER, n_year AS INTEGER
  LOCAL month AS INTEGER, n_month AS INTEGER
  LOCAL day AS INTEGER, n_day AS INTEGER

  LOCAL lMonthChange AS INTEGER
  LOCAL lYearChange AS INTEGER

  '-----
  'default in case of invalid input format causes problems
  NextDay = "YYYYMMDD"

  year = VAL(LEFT$(yyyyymmdd,4))
  month = VAL(MID$(yyyyymmdd,5,2))
  day = VAL(RIGHT$(yyyyymmdd,2))

  'very rough check for valid date
  IF year > 1900 AND year <3000 AND month >0 AND month <13 AND day >0 AND day <32 THEN

    lMonthChange = %FALSE
    lYearChange = %FALSE

    '--- day ---
    SELECT CASE month
      CASE 1, 3, 5, 7, 8, 10, 12
        IF day <31 THEN
          n_day = day + 1
        ELSE
          n_day = 1
          lMonthChange = %TRUE
        END IF
      CASE 4, 6, 9, 11
        IF day <30 THEN
          n_day = day + 1
        ELSE
          n_day = 1
          lMonthChange = %TRUE
        END IF
    END SELECT
  END IF
-----

```

```

        END IF

        CASE 2
        IF year/4 = INT(year/4) THEN 'leap year
        IF day <29 THEN
            n_day = day + 1
        ELSE
            n_day = 1
            lMonthChange = %TRUE
        END IF
        ELSE
        IF day <28 THEN
            n_day = day + 1
        ELSE
            n_day = 1
            lMonthChange = %TRUE
        END IF
        END IF

    END SELECT

    '--- month ---
    IF lMonthChange = %TRUE THEN

        IF month = 12 THEN
            n_month = 1
            lYearChange = %TRUE
        ELSE
            n_month = month + 1
        END IF
    ELSE
        n_month = month
    END IF

    '--- year ---
    IF lYearChange = %TRUE THEN
        n_year = year + 1
    ELSE
        n_year = year
    END IF

    NextDay = TRIM$(STR$(n_year)) + TRIM$( REPEAT$(2-LEN(TRIM$(STR$(n_month))), "0" ) + TRIM$(STR$(n_month))) + _
              TRIM$( REPEAT$(2-LEN(TRIM$(STR$(n_day))), "0" ) + TRIM$(STR$(n_day))) )
END IF

END FUNCTION 'NextDay()

'-----

SUB Flight_DF_Factor(BYVAL FileName AS STRING)

'fill relative duty free factor for each known flight number

CALL LogEntry(FUNCNAME$, "START: " + FileName)

DIM OpsDataLine AS STRING
DIM FieldValue AS STRING
DIM LineField(1 TO 36) AS STRING
DIM FieldCounter AS LONG

DIM LineCounter AS LONG
DIM TotalLines AS LONG

LOCAL NumberOfRecords AS LONG
LOCAL lResult AS LONG

DIM i AS LONG 'general purpose loop counter

'-----

OPEN FileName FOR INPUT AS #1
FILESCAN #1, RECORDS TO NumberOfRecords

    OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #2

OPEN FILE_ERR_08 FOR APPEND AS #4
PRINT #4, "ENTRY: " + DATE$ + " / " + TIME$ + " / " + FileName

OPEN FILE_DF_Flight FOR INPUT AS #3
FILESCAN #3, RECORDS TO TotalLines
DIM aDF(TotalLines,2) AS STRING

'START: read all DF (flight) data into array for later use -----

LineCounter = 1

WHILE NOT EOF(3)

    LINE INPUT #3, OpsDataLine

    IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file

```

```

aDF(LineCounter,1) = EXTRACT$(OpsDataLine, ANY ";") 'get flight number
aDF(LineCounter,1) = LEFT$(aDF(LineCounter,1), LEN(aDF(LineCounter,1))-4) & "0" &
RIGHT$(aDF(LineCounter,1),4) 'insert the "0" for format reason
aDF(LineCounter,2) = RIGHT$(OpsDataLine, (LEN(OpsDataLine) - INSTR(OpsDataLine, ANY ";")) ) 'get DF factor

'---> e.g. [1] (1)AB4916 (2)47,1
INCR LineCounter
END IF

WEND 'EOF(3)

CLOSE #3

'END: read all DF (flight) data into array for later use -----
'-----

OPEN FILE_DF_Country FOR INPUT AS #3
FILESCAN #3, RECORDS TO TotalLines
DIM aDFCountry(TotalLines,3) AS STRING

'START: read all DF (country) data into array for later use -----
LineCounter = 1
WHILE NOT EOF(3)

    LINE INPUT #3, OpsDataLine

    IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file
        aDFCountry(LineCounter,1) = PARSE$(OpsDataLine, ";", 1)
        aDFCountry(LineCounter,2) = PARSE$(OpsDataLine, ";", 2)
        aDFCountry(LineCounter,3) = PARSE$(OpsDataLine, ";", 3)
        '---> e.g. [1] (1)DE (2)GERMANY (3)77,4
        '---> e.g. [2] (1)FR (2)FRANCE (3)93,5
        '...
        INCR LineCounter
    END IF

WEND 'EOF(3)

CLOSE #3

'END: read all DF (country) data into array for later use -----
'-----

'and now fill the org data file

'choose to update only empty fields or to update all fields with new values

lResult = ConsoleMessageBox("Update empty DF factors only [YES] or update/overwrite current content of field [NO]
?", %YESNO+%HANDBOX+%DEFBUTTON1, "WARNING", %IDI_QUESTION, %FALSE)

ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing "+TRIM$(USING$(###,###,###",
NumberOfRecords))+ " Records.", "Reading file...", %FALSE

LineCounter = 1
WHILE ISFALSE EOF(1)

    LINE INPUT #1, OpsDataLine

    'parse the OpsDataLine and fill the field variables

    FieldValue = ""
    FieldCounter = 1

    ProgressBoxUpdate INT(LineCounter/NumberOfRecords*100)

    FOR i = 1 TO LEN(OpsDataLine)
        IF MID$(OpsDataLine, i,1) <> ";" THEN
            'still in same field
            FieldValue = FieldValue + MID$(OpsDataLine, i, 1)
        ELSE
            'field changes, so fill content into LineFieldArray
            LineField(FieldCounter) = FieldValue
            INCR FieldCounter
            FieldValue = ""
        END IF
    NEXT i

    'in case choice was to update/overwrite current content of field or
    'there are still error entries (division by zero) from imported EXCEL source file

    IF lResult = %NOBUTTON OR LineField(%FIELD_FlightPAXDFfactor) = "#DIV/0!" THEN
        'get it from flight number reference
        LineField(%FIELD_FlightPAXDFfactor) = GetDutyFreeFactor(LineField(%FIELD_FlightNumber),
aDF())

    END IF

    'if no DF factor has been found for flight number then use DF country data
    IF LineField(%FIELD_FlightPAXDFfactor) = "XXX" THEN

```



```

aDFcountry()
    LineField(%FIELD_FlightPAXDFfactor) = GetDFCountryFactor(LEFT$(LineField(27),2),
    PRINT #4, "[Check 1: Country DF factor used] " + JOIN$(LineField(), ";")
    END IF

    'if still no DF factor has been found write additionally into error file
    IF LineField(%FIELD_FlightPAXDFfactor) = "XXX" THEN
        PRINT #4, "[Check 2: No DF factor found] " + JOIN$(LineField(), ";")
    END IF

    'write updated file

    OpsDataLine = ""
    FOR i = 1 TO 36
        OpsDataLine = OpsDataLine + TRIM$(LineField(i)) + ";"
    NEXT i

    'omit flights: LH 02989, LH 09998, ZZ-Flights (because of being dummy flights)

    IF LineField(%FIELD_FlightNumber) <> "LH 02989" AND LineField(%FIELD_FlightNumber) <> "LH 09998" AND
    LEFT$(LineField(%FIELD_FlightNumber), 3) <> "ZZ " THEN
        PRINT #2, OpsDataLine
    END IF

    INCR LineCounter

WEND 'EOF(1)

ProgressBoxHide

CLOSE #1
CLOSE #2
CLOSE #4

    ConsoleMessageBox "Check on ErrorQueue_08. It MAY have been updated.",%DEFAULT,"INFO",%DEFAULT,0'
    CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))
    CALL LogEntry(FUNCNAME$, "END: " + FileName)
END SUB 'Flight_DF_Factor()

'-----

FUNCTION GetDutyFreeFactor(BYVAL FlightNo AS STRING, BYREF aDFTable() AS STRING) AS STRING

    DIM i AS LONG
    DIM UpperArrayBoundary AS LONG

    UpperArrayBoundary = UBOUND(aDFTable,1)
    FlightNo = REMOVE$(FlightNo, " ")

    GetDutyFreeFactor = "XXX" 'default value

    FOR i = 1 TO UpperArrayBoundary

        IF FlightNo = REMOVE$(aDFTable(i,1), " ") THEN
            '--> so the flight numbers match
            GetDutyFreeFactor = aDFTable(i,2) 'return the Duty Free factor
            i= UpperArrayBoundary 'to exit the loop
        END IF

    NEXT i

END FUNCTION 'GetDutyFreeFactor

'-----

FUNCTION GetDFCountryFactor(BYVAL Country AS STRING, BYREF aDFcountry() AS STRING) AS STRING

    'scans array of DF factors per country for a specific country and returns DF factor
    'if no match has been found "XXX" is returned

    DIM i AS LONG

    GetDFCountryFactor = "XXX"

    FOR i = 1 TO UBOUND(aDFcountry)

        IF TRIM$(aDFcountry(i,1)) = TRIM$(Country) THEN
            GetDFCountryFactor = aDFcountry(i,3) 'return DF factor
        END IF

    NEXT i

END FUNCTION 'GetDFCountryFactor()

```

```

-----
SUB FillRetailArea(BYVAL FileName AS STRING, BYVAL PurposeIndicator AS INTEGER)

    CALL LogEntry(FUNCNAME$, "START: " + FileName)

'depending on gate fill the actual retail area that a DEP flight belongs to

'----- DECLARATIONS -----
DIM OpsDataLine AS STRING

    DIM Field_FLTNO AS STRING
    DIM Field_COUNTRY AS STRING
    DIM Field_GATE AS STRING
    DIM Field_CKIHALL AS STRING

    DIM Field_RETAILAREA AS STRING

    LOCAL LineCounter AS LONG
    LOCAL TotalLines AS LONG

    LOCAL lResult AS LONG

    LOCAL GatesFound AS LONG
    LOCAL GatesNotFound AS LONG

'----- ROUTINE -----

CALL ShowWaitBox(1)

OPEN FileName FOR INPUT AS #1
    OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #2

'first count number of defined retail areas and then define array and read them

OPEN FILE_RETAIL_AREA_DEF FOR INPUT AS #3

LineCounter = 1
WHILE NOT EOF(3)
    LINE INPUT #3, OpsDataLine

    IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file
        INCR LineCounter
    END IF
WEND 'eof(3)
CLOSE #3

DIM aRetailArea(1 TO LineCounter) AS STRING

'now read areas
OPEN FILE_RETAIL_AREA_DEF FOR INPUT AS #3
LineCounter = 1
WHILE NOT EOF(3)
    LINE INPUT #3, OpsDataLine
    IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file
        aRetailArea(LineCounter) = OpsDataLine
        INCR LineCounter
    END IF
WEND 'eof(3)
CLOSE #3

'read gate info

OPEN FILE_ERR FOR APPEND AS #99
PRINT #99, REPEAT$(LEN(FileName)+24+3, "-")
PRINT #99, "Actual gates entry count: " + FileName
PRINT #99, DATE$ + " / " + TIME$

FILESKAN #1, RECORDS TO TotalLines
ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing "+TRIM$(USING$( "###,###,###",
TotalLines))+ " Records.", "Reading file...", %FALSE

LineCounter = 1

GatesFound = 0
GatesNotFound = 0

WHILE ISFALSE EOF(1)

    LINE INPUT #1, OpsDataLine

'determine retail area

    Field_FLTNO                = PARSE$(OpsDataLine, ";", %FIELD_FlightNumber)
    Field_COUNTRY              = PARSE$(OpsDataLine, ";", %FIELD_DestCountry)
    Field_CKIHALL              = PARSE$(OpsDataLine, ";", %FIELD_CKIHall)

```

```

SELECT CASE PurposeIndicator
CASE 1
    Field_GATE           = PARSE$(OpsDataLine, ";", %FIELD_Gate_Actual)
CASE 2
    Field_GATE           = PARSE$(OpsDataLine, ";", %FIELD_Gate_Season)
CASE 3
    Field_GATE           = PARSE$(OpsDataLine, ";", %FIELD_Gate_Opti)
CASE ELSE
    ConsoleMessageBox "THIS MESSAGE SHOULD NOT APPEAR DURING RUNTIME: WRONG PURPOSEINDICATOR!!!", _
        %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
    CLOSE #99
    CLOSE #2
    CLOSE #1
    EXIT SUB
END SELECT

Field_RETAILAREA = GetRetailArea(aRetailArea(), Field_FLTNO, Field_COUNTRY, Field_GATE, Field_CKIHALL,
GatesFound, GatesNotFound)

SELECT CASE PurposeIndicator
CASE 1
    OpsDataLine = StringUpdate(OpsDataLine, %FIELD_RetailAreaActual, Field_RETAILAREA)
CASE 2
    OpsDataLine = StringUpdate(OpsDataLine, %FIELD_RetailAreaSeason, Field_RETAILAREA)
CASE 3
    OpsDataLine = StringUpdate(OpsDataLine, %FIELD_RetailAreaOpti, Field_RETAILAREA)
END SELECT

ProgressBarUpdate INT(LineCounter/TotalLines*100)

PRINT #2, OpsDataLine
INCR LineCounter

WEND 'EOF(1)

PRINT #99, "Gates found= "; TRIM$(USING$("###,###,###", GatesFound))
PRINT #99, "Gates not found= "; TRIM$(USING$("###,###,###", GatesNotFound))

'-----
CLOSE #99
'-----

CLOSE #1
CLOSE #2

ProgressBarHide

CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))

lResult = ConsoleMessageBox("You should update retail area FACTOR now. Proceed ?", %YESNO+%HANDBOX+%DEFBUTON1,
"WARNING", %IDI_QUESTION, %FALSE)
IF lResult = %NOBUTON THEN
    CALL LogEntry(FUNCNAME$, "NO UPDATE ON RETAIL AREA FACTOR CHOSEN " + FileName)
    ConsoleMessageBox "Data will be inconsistent, if you do not update manually then.\n\nDATA->DATA CLEANING-
>UPDATE RETAIL AREA FACTOR", _
        %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
    ELSE
        CALL LogEntry(FUNCNAME$, "AUTOMATIC UPDATE ON RETAIL AREA FACTOR CHOSEN " + FileName)
        cls : PRINT "Processing season..."
        CALL FillRetailAreaFactor(FileName, PurposeIndicator) 'current working season
        ConsoleMessageBox "Retail areas factors have been filled into data set.",%DEFAULT,"INFO",%DEFAULT,0
        cls
    END IF

    CALL LogEntry(FUNCNAME$, "END: " + FileName)
END SUB 'FillRetailArea()

'-----

FUNCTION GetRetailArea( BYREF RetailAreaDef() AS STRING, BYVAL FltNo AS STRING, BYVAL FltDestCountry AS STRING, _
    BYVAL Gate AS STRING, BYVAL CKI_hall AS STRING, BYREF GatesFound AS LONG , BYREF GatesNotFound
AS LONG) AS STRING

'determine retail area by given gate info

'Retail Areas (defined in File FILE_RETAIL_AREA_DEF)
'-----
'e.g.
'R1= A1-65
'R2= B1-19 'not in transit
'R3= B20-59 'transit
'R4= C (all gates)
'R5= D1-31
'R6= D40-54 'spare bus gates
'R7= E1-26

```

```

LOCAL i AS LONG

IF Gate <> "" THEN 'a gate exists

'default value (in case of non-matching gate name)
GetRetailArea = "RX"

FOR i = 1 TO UBOUND(RetailAreaDef)

        IF INSTR(1, RetailAreaDef(i), Gate) > 0 THEN
                GetRetailArea = LEFT$(RetailAreaDef(i),2)
                INCR GatesFound
                i = UBOUND(RetailAreaDef)
        END IF

NEXT i

ELSE 'no gate exists

INCR GatesNotFound

'default value (in case of empty gate name)
GetRetailArea = "RY"

'but in case CheckIn hall is known then allocate as follows
IF CKI_hall = "C" THEN
        GetRetailArea = "R4"

ELSEIF CKI_hall = "D" THEN

        IF LEFT$(FltNo, 2) = "BA" THEN
                GetRetailArea = "R6"
        ELSE
                GetRetailArea = "R5"
        END IF

ELSEIF CKI_hall = "E" THEN
        GetRetailArea = "R7"

ELSE 'CKI hall A or B

        IF Is_EU_flight(FltDestCountry) = %TRUE THEN

                IF LEFT$(FltNo, 2) = "LH" THEN
                        'in case of Lufthansa flight, most probably: finger A
                        GetRetailArea = "R1" 'Lufthansa
                ELSE
                        GetRetailArea = "R2" 'non-transit (non-LH, EU)
                END IF

                ELSE 'non-EU
                        GetRetailArea = "R3" 'transit
                END IF

        END IF

END IF ' Gate <> ""

END FUNCTION 'GetRetailArea

'-----

FUNCTION Is_EU_flight(BYVAL FltDestCountry AS STRING) AS INTEGER

SELECT CASE LEFT$(FltDestCountry, 2)

CASE "AT", "BE", "BG", "CY", "CZ", "DK", "EE", "FI", "FR", _
"DE", "GR", "HU", "IE", "IT", "LV", "LT", "LU", "MT", _
"NL", "PL", "PT", "RO", "SI", "ES", "SE", "GB"

        Is_EU_flight = %TRUE

CASE ELSE

        Is_EU_flight = %FALSE 'means: it is non-EU

END SELECT

END FUNCTION 'Is_EU_flight(FltDestCountry)

'-----

SUB FillRetailAreaFactor(BYVAL FileName AS STRING, BYVAL PurposeIndicator AS INTEGER)

CALL LogEntry(FUNCNAME$, "START")

```

```

'fills retail area factor dependend on actual retail area
'the factor is drawn from a table, indicating the relative performance
'of each retail area compared to average

'----- DECLARATIONS -----

DIM OpsDataLine AS STRING
DIM LineCounter AS LONG
DIM TotalLines AS LONG
LOCAL ProgressTotal AS LONG

DIM i AS LONG 'general purpose loop counter

'----- ROUTINE -----

OPEN FILE_RetailAreaFactors FOR INPUT AS #1

'open season origin file
  OPEN FileName FOR INPUT AS #2
  FILESCAN #2, RECORDS TO ProgressTotal

  'open new target file
  OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #3

FILESCAN #1, RECORDS TO TotalLines
DIM aRF(TotalLines,2) AS STRING

'START: read all RF data into array for later use -----

LineCounter = 1
WHILE NOT EOF(1)

  LINE INPUT #1, OpsDataLine

  IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file
    aRF(LineCounter,1) = LEFT$(OpsDataLine, 2)
    aRF(LineCounter,2) = RIGHT$(OpsDataLine, (LEN(OpsDataLine) - INSTR(OpsDataLine, ANY ";")) ) 'get RF factor
    INCR LineCounter
  END IF

WEND 'EOF(1)

CLOSE #1

'END:  read all RF data into array for later use -----

LineCounter = 1
ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing "+TRIM$(USING$(###,###,###",
ProgressTotal))+ " Records.", "Reading file...", %FALSE
WHILE ISFALSE EOF(2)

  LINE INPUT #2, OpsDataLine

  SELECT CASE PurposeIndicator

    CASE 1
      OpsDataLine = StringUpDate(OpsDataLine, %FIELD_RetailAreaFactorActual,
GetRetailFactor(PARSE$(OpsDataLine, ";", %FIELD_RetailAreaActual), aRF()))
    CASE 2
      OpsDataLine = StringUpDate(OpsDataLine, %FIELD_RetailAreaFactorSeason,
GetRetailFactor(PARSE$(OpsDataLine, ";", %FIELD_RetailAreaSeason), aRF()))
    CASE 3
      OpsDataLine = StringUpDate(OpsDataLine, %FIELD_RetailAreaFactorOpti,
GetRetailFactor(PARSE$(OpsDataLine, ";", %FIELD_RetailAreaOpti), aRF()))

    CASE ELSE

      ConsoleMessageBox "THIS MESSEGE SHOULD NOT APPEAR DURING RUNTIME: WRONG PURPOSEINDICATOR!!!", _
        %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0

      CLOSE #3
      CLOSE #2
      EXIT SUB

  END SELECT

  'write updated file
  PRINT #3, OpsDataLine

  INCR LineCounter
  ProgressBoxUpdate INT(LineCounter/ProgressTotal*100)

WEND 'EOF(2)

ProgressBoxHide

CLOSE #2
CLOSE #3

```

```

        CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))
        CALL LogEntry(FUNCNAME$, "END")

        cls
    PRINT
    PRINT "Cleaning up..."
    PRINT

END SUB 'FillRetailAreaFactor()

'-----

FUNCTION GetRetailFactor(BYVAL RetailArea AS STRING, BYREF aRFTable() AS STRING) AS STRING

    DIM i AS LONG
    DIM UpperArrayBoundary AS LONG

    UpperArrayBoundary = UBOUND(aRFTable,1)
    RetailArea = REMOVE$(RetailArea, " ")

    GetRetailFactor = "XXX"           'default value

    FOR i = 1 TO UpperArrayBoundary

        IF RetailArea = REMOVE$(aRFTable(i,1), " ") THEN
            '--> so the retail area matches
            GetRetailFactor = aRFTable(i,2) 'return the retail area factor
            i= UpperArrayBoundary         'to exit the loop
        END IF
    NEXT i

END FUNCTION 'GetRetailFactor

'-----

SUB FillErrorQueue(BYVAL FileName AS STRING)

    '----- DECLARATIONS -----
    DIM OpsDataLine AS STRING

    '----- ROUTINE -----

    'open season origin file
    OPEN FileName FOR INPUT AS #1

    'open output file (error queue)
    OPEN FILE_ERR FOR APPEND AS #2

    WHILE ISFALSE EOF(1)

        LINE INPUT #1, OpsDataLine

        'parse the OpsDataLine and fill the field variables

        '----- determine specific error situations -----

        IF          PARSE$(OpsDataLine, ";", %FIELD_PAX_Actual) = "" THEN
            'write into ErrFile
            PRINT #2, DATE$ + ";" + TIME$ + ";" + OpsDataLine
        END IF

    WEND 'EOF(1)

    CLOSE #1
    CLOSE #2

    ConsoleMessageBox "Error queue appended: " + FILE_ERR,%DEFAULT,"INFO",%DEFAULT,0

END SUB 'FillErrorQueue()

'-----

SUB UpdateOAGFile(hWndForm AS DWORD)

```

```

CALL LogEntry(FUNCNAME$, "START")

' uses OrgDataFile to determine the destination airport (first a/p after FRA)
' write new OAG_File

' assumes that info is valid for both summer and winter season

' after this sub routine has run, LoadOpsDataCountry and FillRetailArea needs TO run again

'----- DECLARATIONS -----

DIM OpsDataLine AS STRING
DIM OAGDataLine AS STRING
DIM UniqueFlightNumbers (1 TO 15000) AS STRING
DIM ArrayPointer AS LONG
DIM IsInArray AS INTEGER
DIM i AS LONG

DIM lResult AS LONG

'----- ROUTINE -----

OPEN FILE_ORG FOR INPUT AS #1
OPEN FILE_OAG FOR OUTPUT AS #2

ArrayPointer = 1

WHILE ISFALSE EOF(1)

    LINE INPUT #1, OpsDataLine

    'parse the OpsDataLine and construct OAGDataLine

    OAGDataLine = MID$(OpsDataLine, 9, 8) + SPACE$(8) + MID$(OpsDataLine, 606, 3)

    'if not yet in OAG_File

    IsInArray = 0

    FOR i = 1 TO ArrayPointer 'test each element in array

        IF UniqueFlightNumbers(i) = MID$(OpsDataLine, 9, 8) THEN
            'is already in array
            i = ArrayPointer 'to exit the loop
            IsInArray = 1
        END IF

    NEXT i

    IF IsInArray = 0 THEN ' if not yet in array
        'insert into array
        ARRAY INSERT UniqueFlightNumbers(), MID$(OpsDataLine, 9, 8)
        INCR ArrayPointer
        'write into OAG_File
        PRINT #2, OAGDataLine
    END IF

WEND 'EOF(1)

CLOSE #1
CLOSE #2

'after this sub routine has run, LoadOpsDataCountry and FillRetailArea needs TO run again

lResult = ConsoleMessageBox("Country and retail info needs to be updated. Update now?",
%YESNO+%HANDBOX+%DEFBUTTON1, "WARNING", %IDI_QUESTION, %FALSE)

IF lResult = %YESBUTTON THEN

    'update country info
    ConsoleMessageBox "country: summer season",%DEFAULT,"INFO",%DEFAULT,0
    ConsoleMessageBox "country: winter season",%DEFAULT,"INFO",%DEFAULT,0

    'update retail area
    ConsoleMessageBox "retail area: summer season",%DEFAULT,"INFO",%DEFAULT,0
    ConsoleMessageBox "retail area: winter season",%DEFAULT,"INFO",%DEFAULT,0

    'update retail area factor
    ConsoleMessageBox "retail area factor: summer season",%DEFAULT,"INFO",%DEFAULT,0
    ConsoleMessageBox "retail area factor: winter season",%DEFAULT,"INFO",%DEFAULT,0

    'update retail revenue
    ConsoleMessageBox "retail revenue: summer season",%DEFAULT,"INFO",%DEFAULT,0
    ConsoleMessageBox "retail revenue: winter season",%DEFAULT,"INFO",%DEFAULT,0

ELSE
    ConsoleMessageBox "BE AWARE TO UPDATE MANUALLY THEN.",%DEFAULT,"INFO",%DEFAULT,0
    CALL LogEntry(FUNCNAME$, "no automatic update chosen")
END IF

```

```

CALL LogEntry(FUNCNAME$, "END")
END SUB 'UpdateOAGFile()

-----

SUB GenerateWingSpanCode()

'uses ORG data file to determine different types of A/C and their WingSpanCode
'----- DECLARATIONS -----
DIM UniqueACType (1 TO 300) AS STRING
DIM OpsDataLine AS STRING
DIM IsInArray AS INTEGER
DIM ArrayPointer AS LONG

DIM i AS LONG

'----- ROUTINE -----

OPEN FILE_ORG FOR INPUT AS #1
OPEN FILE_WSC FOR OUTPUT AS #2

ArrayPointer = 1

WHILE ISFALSE EOF(1)

    LINE INPUT #1, OpsDataLine

    IsInArray = 0

    'if not yet in AC_array

    FOR i = 1 TO ArrayPointer 'test each element in array

        IF LEFT$(UniqueACType(i), 4) = MID$(OpsDataLine, 63, 4) THEN
            'is already in array
            i = ArrayPointer 'to exit the loop
            IsInArray = 1
        END IF

    NEXT i

    IF IsInArray = 0 THEN ' if not yet in array

        'insert into array if there is a valid Wing Span Code
        IF MID$(OpsDataLine, 480, 2) <> " " THEN
            ARRAY INSERT UniqueACType(), MID$(OpsDataLine, 63, 4) + ";" + MID$(OpsDataLine, 480, 2)
            PRINT #2, TRIM$(MID$(OpsDataLine, 63, 4)) + ";" + TRIM$(MID$(OpsDataLine, 480, 2)) ' e.g. "B744:02"
            INCR ArrayPointer
        END IF

    END IF

WEND 'EOF(1)

CLOSE #1
CLOSE #2

END SUB 'GenerateWingSpanCode()

-----

SUB GenerateAvgGroundTime()

CALL LogEntry(FUNCNAME$, "START")

'uses ORG data file to determine different types of A/C and their average standard ground time
'round values to the next 5 minutes and write output file

'----- DECLARATIONS -----
DIM UniqueACType (1 TO 300, 1 TO 3) AS STRING
DIM OpsDataLine AS STRING
DIM IsInArray AS INTEGER
DIM ArrayPointer AS LONG

DIM i AS LONG

'----- ROUTINE -----

OPEN FILE_ORG FOR INPUT AS #1
OPEN FILE_GROUNDTIME FOR OUTPUT AS #2

```



```

ArrayPointer = 1
WHILE ISFALSE EOF(1)
    LINE INPUT #1, OpsDataLine
    IsInArray = 0
    'if not yet in AC_array
    FOR i = 1 TO ArrayPointer 'test each element in array
        IF TRIM$(LEFT$(UniqueACType(i,1), 4)) = TRIM$(MID$(OpsDataLine, 63, 4)) THEN
            'is already in array
            'insert into array if there is a valid Standard Ground Time
            IF LEN(TRIM$(MID$(OpsDataLine, 582, 4))) > 0 AND VAL(TRIM$(MID$(OpsDataLine, 582, 4)))
<> 0 THEN
                UniqueACType(i,2) = STR$(VAL(UniqueACType(i,2)) + VAL(MID$(OpsDataLine, 582, 4))) 'Add
standard ground time to current sum
                UniqueACType(i,3) = STR$(VAL(UniqueACType(i,3)) + 1)
                'increase occurrence by 1
                i = ArrayPointer 'to exit the loop
                IsInArray = 1
            END IF
        END IF
    NEXT i

    IF IsInArray = 0 THEN ' if not yet in array
        'insert into array if there is a valid Standard Ground Time
        IF LEN(TRIM$(MID$(OpsDataLine, 582, 4))) > 0 AND VAL(TRIM$(MID$(OpsDataLine, 582, 4))) <> 0 THEN
            ARRAY INSERT UniqueACType(1,1), TRIM$(MID$(OpsDataLine, 63, 4)) ' A/C type
            ARRAY INSERT UniqueACType(1,2), TRIM$(MID$(OpsDataLine, 582, 4)) ' Standard Ground
Time
            ARRAY INSERT UniqueACType(1,3), "1"
            ' First occurrence
            INCR ArrayPointer
        END IF
    END IF

    WEND 'EOF(1)

    FOR i = 1 TO ArrayPointer
        IF VAL(UniqueACType(i,3)) <> 0 AND VAL(UniqueACType(i,2)) <> 0 THEN
            PRINT #2, UniqueACType(i,1) + ";" + TRIM$(STR$(
INT((ROUND((VAL(UniqueACType(i,2)) / VAL(UniqueACType(i,3))), 0)+4)/5)*5 ))
        END IF
    NEXT i

    CLOSE #1
    CLOSE #2

    CALL LogEntry(FUNCNAME$, "END")
END SUB 'GenerateAvgGroundTime()

'-----
SUB FillGroundTime(BYVAL FileName AS STRING)
    CALL LogEntry(FUNCNAME$, "START")
    'fills the standard ground time field for records that are empty at that field
    'data source is a computed avg ground time taken from an ABA orginal data set

    '----- DECLARATIONS -----
    DIM TotalLines AS LONG
    DIM OpsDataLine AS STRING
    DIM LineCounter AS LONG

    '----- ROUTINE -----
    OPEN FileName FOR INPUT AS #1
    OPEN FILE_GroundTime FOR INPUT AS #2
    OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #3
    'read all avg ground times into an array -----
    FILESCAN #2, RECORDS TO TotalLines
    DIM aGroundTimes(1 TO TotalLines+1, 1 TO 2) AS STRING

    LineCounter = 1
    WHILE NOT EOF(2)
        LINE INPUT #2, OpsDataLine
        aGroundTimes(LineCounter, 1) = PARSE$(OpsDataLine, ":", 1)
    
```

```

        aGroundTimes(LineCounter, 2) = PARSE$(OpsDataLine, ";", 2)
        INCR LineCounter

    WEND 'EOF(2)

' test for each record in OrgData if standard ground time is filled
' in case not fill it with a value from aGroundTimes Array

LineCounter = 0

    WHILE NOT EOF(1)

        LINE INPUT #1, OpsDataLine

        'if there is no standard ground time (field 25)
        IF VAL(PARSE$(OpsDataLine, ";", %FIELD_StdGroundTime)) = 0 THEN

            'look up A/C type in ground time array, store position in LineCounter
            ARRAY SCAN aGroundTimes(), =PARSE$(OpsDataLine, ";", %FIELD_ACType), TO LineCounter

            'if A/C type found
            IF LineCounter <> 0 THEN
                'take/fill avg ground time from array as standard ground time
                OpsDataLine = StringUpdate(OpsDataLine, %FIELD_StdGroundTime,
aGroundTimes(LineCounter,2))
            ELSE
                ConsoleMessageBox "No entry for STANDARD GROUND TIME found !!!" + _
                    "/nAC type: " + PARSE$(OpsDataLine, ";",
%FIELD_ACType),%DEFAULT,"INFO",%DEFAULT,0
            END IF

            ELSE
                'reformat ground time (eliminate leading zeros)
                OpsDataLine = StringUpdate(OpsDataLine, %FIELD_StdGroundTime,
TRIM$(STR$(VAL(PARSE$(OpsDataLine, ";", %FIELD_StdGroundTime))))
            END IF

            PRINT #3, OpsDataLine

        WEND 'EOF(1)

        CLOSE #1
        CLOSE #2
        CLOSE #3

        CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))

        CALL LogEntry(FUNCNAME$, "END")

    END SUB 'FillGroundTime()

'-----
SUB GenerateGatesUsed()

    CALL LogEntry(FUNCNAME$, "START")

'uses ORG data file to determine different gates

'----- DECLARATIONS -----

DIM UniqueGateType (1 TO 1000) AS STRING
DIM OpsDataLine AS STRING
DIM IsInArray AS INTEGER
DIM ArrayPointer AS LONG

DIM i AS LONG 'general purpose loop counter

'----- ROUTINE -----

OPEN FILE_ORG FOR INPUT AS #1
OPEN FILE_GATE FOR OUTPUT AS #2

ArrayPointer = 1

WHILE ISFALSE EOF(1)

    LINE INPUT #1, OpsDataLine

    IsInArray = 0

    'if not yet in AC_array

    FOR i = 1 TO ArrayPointer 'test each element in array

        IF LEFT$(UniqueGateType(i),4) = MID$(OpsDataLine, 361, 4) THEN
            'is already in array
            i = ArrayPointer 'to exit the loop
            IsInArray = 1
        END IF
    END IF

```

```

NEXT i

IF IsInArray = 0 THEN ' if not yet in array

    ARRAY INSERT UniqueGateType(), MID$(OpsDataLine, 361, 4) + ";000"
    PRINT #2, MID$(OpsDataLine, 361, 4) + ";000" 'e.g. "A13;000"
    INCR ArrayPointer

END IF

WEND 'EOF(1)

CLOSE #1
CLOSE #2

CALL LogEntry(FUNCNAME$, "END")

END SUB 'GenerateGatesUsed()

'-----
SUB FillFlightRevenueSpecific(BYVAL FileName AS STRING, BYVAL PurposeIndicator AS INTEGER)

    CALL LogEntry(FUNCNAME$, "START")

    'calculates the retail revenue for each flight

    'formula:
    '      retail revenue per flight = NumberOfPax      * AvgDFRevenue      * (FlightPaxDFfactor/100)      *
DF->RetailFactor      * RetailAreaFactor
    '      = %FIELD_PAX_Actual * RevPerPax      * (%FIELD_FlightPAXDFfactor / 100) *
FILE_DF_RETAIL_FACTOR * %FIELD_RetailAreaFactor[Actual|Season|Opti]
    '      =      233      *      0,80      *      (121,6/100)      *
3,4      *      2,7
    '      = 2600,95

    '----- DECLARATIONS -----
DIM OpsDataLine AS STRING

DIM Revenue AS STRING
DIM DF_RetailFactor AS STRING
DIM RevPerPax AS CUR
DIM FlightPaxDFfactor AS STRING
DIM RetailAreaFactor AS STRING
DIM LineCounter AS LONG
DIM TotalLines AS LONG

    '----- ROUTINE -----

OPEN FILE_DF_RETAIL_FACTOR FOR INPUT AS #1      'File with DF->RetailFactor value

'open season origin file
OPEN FileName FOR INPUT AS #2
FILESIZE #2, RECORDS TO TotalLines

'open new target file
OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #3

'START: read DF->RetailFactor for later use -----
--

WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    'the first non-comment-line in file is to be the DF_RetailFactor value
    IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file

        REPLACE ",," WITH "." IN OpsDataLine 'just in case a comma instead of decimal point is used ( "3,40" -->
"3.40")

        DF_RetailFactor = OpsDataLine

    END IF

WEND 'EOF(1)

CLOSE #1

'END: read DF->RetailFactor later use -----
--

OPEN FILE_RevPerPax FOR INPUT AS #1      'File with Average (DF) Revenue Per PAX

'START: read avg. PAX-DF-Revenue for later use -----
--

WHILE NOT EOF(1)

```

```

LINE INPUT #1, OpsDataLine

'the first non-comment-line in file is to be the RevPerPax value
IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file

    REPLACE "," WITH "." IN OpsDataLine 'just in case a comma instead of decimal point is used ( "5,80" -->
"5.80")

    RevPerPax = VAL(OpsDataLine)

END IF

WEND 'EOF(1)

CLOSE #1

'END: read avg. PAX-DF-Revenue for later use -----
-----

ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing "+TRIM$(USING$( "###,###,###",
TotalLines))+ " Records.", "Calculating Revenue...", %FALSE
LineCounter = 1
WHILE ISFALSE EOF(2)

    LINE INPUT #2, OpsDataLine

    'convert FlightPaxDFfactor into a decimal value format
    FlightPaxDFfactor = PARSE$(OpsDataLine, ";", %FIELD_FlightPAXDFfactor)
    REPLACE "," WITH "." IN FlightPaxDFfactor 'just in case a comma instead of decimal point is used ( "121,6" -->
"121.6")

    SELECT CASE PurposeIndicator
    CASE 1
        RetailAreaFactor = PARSE$(OpsDataLine, ";", %FIELD_RetailAreaFactorActual)
    CASE 2
        RetailAreaFactor = PARSE$(OpsDataLine, ";", %FIELD_RetailAreaFactorSeason)
    CASE 3
        RetailAreaFactor = PARSE$(OpsDataLine, ";", %FIELD_RetailAreaFactorOpti)
    CASE ELSE
        ConsoleMessageBox "THIS MESSEGE SHOULD NOT APPEAR DURING RUNTIME: WRONG PURPOSEINDICATOR!!!", _
            %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

        CLOSE #3
        CLOSE #2
        EXIT SUB
    END SELECT

    'convert RetailAreaFactor into a decimal value format
    REPLACE "," WITH "." IN RetailAreaFactor 'just in case a comma instead of decimal point is used ( "2,7" -->
"2.7")

    'formula:
    '
    '      retail revenue per flight = NumberOfPax      * AvgDFRevenue      * (FlightPaxDFfactor/100)
    * DF->RetailFactor      * RetailAreaFactor
    '
    '      = %FIELD_PAX_Actual * RevPerPax      * (%FIELD_FlightPAXDFfactor /
100) * FILE_DF_RETAIL_FACTOR * %FIELD_RetailAreaFactor[Actual|Season|Opti]
    '
    '      = 233      * 0,80      * (121,6/100)
*
    '      3,4      *      2,7
    '
    '      = 2600,95

    'calculate revenue

    IF VAL(RetailAreaFactor) <> 0 THEN 'for the case SEASON or OPTI is chosen and values of "XXX" or similar are
found
        Revenue = TRIM$(STR$( ROUND ( _
            RevPerPax * _
            VAL(PARSE$(OpsDataLine, ";", %FIELD_PAX_Actual) ) * VAL(FlightPaxDFfactor) / 100 *
            VAL(DF_RetailFactor) * _
            VAL(RetailAreaFactor) _
            ,2)))

        'convert Revenue back into a comma value format
        'REPLACE "." WITH "," IN Revenue      ' e.g. "121.6" --> "121,6")
    ELSE
        Revenue = "XXX"
    END IF

    SELECT CASE PurposeIndicator
    CASE 1
        OpsDataLine = StringUpDate(OpsDataLine, %FIELD_RetailRevenueActual, Revenue)
    CASE 2
        OpsDataLine = StringUpDate(OpsDataLine, %FIELD_RetailRevenueSeason, Revenue)
    CASE 3
        OpsDataLine = StringUpDate(OpsDataLine, %FIELD_RetailRevenueOpti, Revenue)
    END SELECT

```

```

        ProgressBoxUpdate INT(LineCounter/TotalLines*100)
        INCR LineCounter

        PRINT #3, OpsDataLine

    WEND 'EOF(2)

    CLOSE #2
    CLOSE #3

    ProgressBoxHide

        CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))
        CALL LogEntry(FUNCNAME$, "END")
END SUB 'FillFlightRevenueSpecific()

'-----
SUB ReIndex(BYVAL FileName AS STRING)

    CALL LogEntry(FUNCNAME$, "START")

    '(re-)index the data file

    '----- DECLARATIONS -----
    DIM OpsDataLine AS STRING
    DIM LineCounter AS LONG
    DIM FileNameText AS STRING

    DIM lResult AS LONG

    '----- ROUTINE -----
    ' IF LEN(FileName) > 20 THEN
    '     FileNameText = RIGHT$(FileName,20)
    ' ELSE
    '     FileNameText = FileName
    ' END IF

    lResult = ConsoleMessageBox("Are you sure to re-index file: " + TRIM$(FileNameText) + " ?",
    %YESNO+%HANDBOX+%DEFBUTTON2, "WARNING", %IDI_QUESTION, %FALSE)

    IF lResult = %YESBUTTON THEN

        'open season origin file
        OPEN FileName FOR INPUT AS #1

        'open new target file
        OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #2

        LineCounter = 1
        WHILE NOT EOF(1)

            LINE INPUT #1, OpsDataLine
            PRINT #2, StringUpdate(OpsDataLine, %FIELD_RecID, TRIM$(STR$(LineCounter)))
            INCR LineCounter

        WEND 'EOF(1)

        CLOSE #1
        CLOSE #2

        ConsoleMessageBox "File re-indexed.",%DEFAULT,"INFO",%DEFAULT,0

        CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))

    ELSE
        ConsoleMessageBox "Nothing re-indexed.",%DEFAULT,"INFO",%DEFAULT,0
    END IF

    CALL LogEntry(FUNCNAME$, "END: " + FileName)
END SUB 'ReIndex()

'-----
SUB ReadAllRecsIntoArray(BYVAL FileName AS STRING)

    CALL LogEntry(FUNCNAME$, "START: " + FileName)

    '----- DECLARATIONS -----

```

```

DIM OpsDataLine AS STRING
DIM FieldValue AS STRING
DIM FieldCounter AS LONG

DIM TotalLines AS LONG
DIM LineCounter AS LONG

DIM i AS LONG 'general purpose loop counter

DIM lResult AS LONG

'----- ROUTINE -----
'if called for more seasons in one run do not overwrite error files

lResult = ConsoleMessageBox("Overwrite error files?", %YESNO+%HANDBOX+%DEFBUTTON2, "WARNING", %IDI_QUESTION,
%FALSE)

IF lResult = %YESBUTTON THEN

    ConsoleMessageBox "Error files will be
overwritten.",%OKONLY+%EXCLAMATIONBOX,"INFO",%IDI_EXCLAMATION,0

    CALL LogEntry(FUNCNAME$, "overwrite error files")

    OPEN FILE_ERR_03 FOR OUTPUT AS #1
    CLOSE #1

    OPEN FILE_ERR_04 FOR OUTPUT AS #1
    CLOSE #1

    OPEN FILE_ERR_05 FOR OUTPUT AS #1
    CLOSE #1

    OPEN FILE_ERR_06 FOR OUTPUT AS #1
    CLOSE #1

ELSE
    ConsoleMessageBox "Any error record found will be
appended.",%OKONLY+%EXCLAMATIONBOX,"INFO",%IDI_EXCLAMATION,0

    CALL LogEntry(FUNCNAME$, "append to error files")

END IF

CALL ShowWaitBox(1)

OPEN FileName FOR INPUT AS #1

FILESCAN #1, RECORDS TO TotalLines
DIM FlightRec(1 TO TotalLines+1, 1 TO 36) AS STRING

PRINT : PRINT "Number of records to be processed: " + TRIM$(USING$( "###,###,###", TotalLines))

ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing "+TRIM$(USING$( "###,###,###",
TotalLines))+ " Records.", "Reading file...", %FALSE

LineCounter = 1

WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    'parse the OpsDataLine and fill the field variables

    FieldValue = ""
    FieldCounter = 1

    FOR i = 1 TO LEN(OpsDataLine)
        IF MID$(OpsDataLine, i,1) <> ";" THEN
            'still in same field
            FieldValue = FieldValue + MID$(OpsDataLine, i, 1)
        ELSE
            'field changes, so fill content into FlightRec Array
            FlightRec(LineCounter, FieldCounter) = FieldValue
            INCR FieldCounter
            FieldValue = ""
        END IF
    NEXT i

    ProgressBoxUpdate INT(LineCounter/TotalLines*100)

    INCR LineCounter

WEND 'EOF(1)

ProgressBoxHide

CLOSE #1

'----- SOME MORE DATA CLEANING (REPORTS) -----

```

```

'-----
'----- flights with no ATD -----
'-----

CALL LogEntry(FUNCNAME$, "report flights with no ATD into error file 03")
PRINT : PRINT "Checking flights with no ATD..."

OPEN FILE_ERR_03 FOR APPEND AS #1
PRINT #1, "ENTRY: " + DATE$ + " / " + TIME$ + " / " + FileName

FOR i = 1 TO TotalLines

    OpsDataLine = ""

    IF FlightRec(i, 6) = "" THEN
        'no ATD
        CALL WriteRec(FlightRec(), i)
    END IF

NEXT i

CLOSE #1

'-----
'----- flights with no pax actual -----
'-----

CALL LogEntry(FUNCNAME$, "report flight with no actual pax in error file 04")
PRINT "Checking flights with no ACTUAL PAX..."

OPEN FILE_ERR_04 FOR APPEND AS #1
PRINT #1, "ENTRY: " + DATE$ + " / " + TIME$ + " / " + FileName

FOR i = 1 TO TotalLines

    OpsDataLine = ""

    IF FlightRec(i, 19) = "" THEN
        'no actual pax
        CALL WriteRec(FlightRec(), i)
    END IF

NEXT i

CLOSE #1

'-----
'----- flights with no DF PAX factor -----
'-----

CALL LogEntry(FUNCNAME$, "report flights with no DF pax factor into error file 05")
PRINT "Checking flights with no PAX DF factor..."

OPEN FILE_ERR_05 FOR APPEND AS #1
PRINT #1, "ENTRY: " + DATE$ + " / " + TIME$ + " / " + FileName

FOR i = 1 TO TotalLines

    OpsDataLine = ""

    IF FlightRec(i, 26) = "" OR FlightRec(i, 26) = "XXX" THEN
        'no flight pax DF factor
        CALL WriteRec(FlightRec(), i)
    END IF

NEXT i

CLOSE #1

'-----
'----- flights with a delay of "9999" -----
'-----

CALL LogEntry(FUNCNAME$, "report files with a delay of '9999' into error file 06")
PRINT "Checking flights with a delay of '9999'..."

OPEN FILE_ERR_06 FOR APPEND AS #1
PRINT #1, "ENTRY: " + DATE$ + " / " + TIME$ + " / " + FileName

FOR i = 1 TO TotalLines

    OpsDataLine = ""

    IF FlightRec(i, 7) = "9999" THEN
        'unspecified delay minutes
        CALL WriteRec(FlightRec(), i)
    END IF

NEXT i

CLOSE #1

```

```

ConsoleMessageBox "Check on Error Files: \n\n 3 (no ATD) , 4 (no pax actual), \n 5 (no pax DF factor), 6 (delay OF
9999).",%DEFAULT,"INFO",%DEFAULT,0

```

```

'----- SOME SAMPLE STATISTICS -----

'----- list of countries -----
'----- list of airlines -----
'----- avg pax DF factor per country -----
'----- total pax actual -----
'----- total pax booked -----
'----- total pax per gate -----
'----- total pax per retail area -----
'----- total pax per country -----
'----- total pax per airline -----
'----- total revenue -----
'----- total revenue per gate -----
'----- total revenue per retail area -----
'----- avg revenue per gate per pax -----
'----- avg revenue per retail area per pax -----
'----- list with number of gates used per day ---

CALL LogEntry(FUNCNAME$, "END: " + FileName)

cls
PRINT
PRINT "Cleaning up..."
PRINT

END SUB 'ReadAllRecsIntoArray()

'-----

SUB WriteRec(BYREF FlightArray() AS STRING, BYVAL LineCounter AS LONG)

'assumes that file handle #1 is opened for output

DIM FieldCounter AS LONG
DIM OpsDataLine AS STRING

OpsDataLine = ""
FOR FieldCounter = 1 TO 36
    OpsDataLine = OpsDataLine + FlightArray(LineCounter, FieldCounter) + ";"
NEXT FieldCounter

PRINT #1, OpsDataLine

END SUB 'WriteRec(BYREF FlightArray() AS STRING, BYVAL LineCounter AS LONG)

'-----

FUNCTION StringUpdate(BYVAL StringData AS STRING, BYVAL FieldNumber AS INTEGER, BYVAL UpDateText AS STRING) AS STRING

DIM Occurrences AS LONG

Occurrences = PARSECOUNT(StringData, ANY ";")

IF Occurrences > 1 THEN

    DIM aSearch(1 TO Occurrences) AS STRING

    PARSE StringData, aSearch(), ANY ";"
    aSearch(FieldNumber) = UpDateText

    StringUpDate = JOIN$(aSearch(), ";")

ELSE
    StringUpDate = StringData
END IF

END FUNCTION 'StringUpDate()

'-----

SUB FillPlanningSeason()

CALL LogEntry(FUNCNAME$, "START")

'fills OrgDataFile with seasonal planning data
'summer season 2006 and winter season 2006/2007

'seasonal planning data is available for a week (so called reference week)
'the days within reference week have been mapped to whole season

'----- DECLARATIONS -----

```



```

DIM OpsDataLine AS STRING

DIM TotalLines AS LONG
DIM FieldCounter AS INTEGER
DIM FieldValue AS STRING
DIM LineCounter AS LONG

DIM i AS LONG           'general purpose loop counter

DIM OrgDataRec AS LONG 'loop counter
DIM SeasonRec AS LONG  'loop counter

'----- ROUTINE -----
OPEN FILE_ERR_07 FOR OUTPUT AS #2           'error queue that will contain records of
                                           'seasonal flight plan with no matches in OrgDataFile

'----- Summer Season -----

CALL LogEntry(FUNCNAME$, "fill summer seasonal plan data into array")

OPEN FILE_SS FOR INPUT AS #1

FILESCAN #1, RECORDS TO TotalLines
DIM aSummerSeason(1 TO TotalLines+1, 1 TO 7) AS STRING

LineCounter = 1

WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    'parse the OpsDataLine and fill the field variables

    FieldValue = ""
    FieldCounter = 1

    FOR i = 1 TO LEN(OpsDataLine)
        IF MID$(OpsDataLine, i,1) <> ";" THEN
            'still in same field
            FieldValue = FieldValue + MID$(OpsDataLine, i, 1)
        ELSE
            'field changes, so fill content into FlightRec Array
            aSummerSeason(LineCounter, FieldCounter) = FieldValue
            INCR FieldCounter
            FieldValue = ""
        END IF
    NEXT i

    INCR LineCounter

WEND 'EOF(1)

CLOSE #1

'----- Winter Season -----

CALL LogEntry(FUNCNAME$, "fill winter seasonal plan data into array")

OPEN FILE_WS FOR INPUT AS #1

FILESCAN #1, RECORDS TO TotalLines
DIM aWinterSeason(1 TO TotalLines+1, 1 TO 7) AS STRING

LineCounter = 1

WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    'parse the OpsDataLine and fill the field variables

    FieldValue = ""
    FieldCounter = 1

    FOR i = 1 TO LEN(OpsDataLine)
        IF MID$(OpsDataLine, i,1) <> ";" THEN
            'still in same field
            FieldValue = FieldValue + MID$(OpsDataLine, i, 1)
        ELSE
            'field changes, so fill content into FlightRec Array
            aWinterSeason(LineCounter, FieldCounter) = FieldValue
            INCR FieldCounter
            FieldValue = ""
        END IF
    NEXT i

    INCR LineCounter

WEND 'EOF(1)

CLOSE #1

```

```

'----- OrgDataFile SUMMER -----
CALL LogEntry(FUNCNAME$, "fill summer org data into array")

OPEN FILE_SUMMER FOR INPUT AS #1

FILESCAN #1, RECORDS TO TotalLines
DIM aOrgDataFileSummer(1 TO TotalLines+1, 1 TO 36) AS STRING

LineCounter = 1

WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    'parse the OpsDataLine and fill the field variables

    FieldValue = ""
    FieldCounter = 1

    FOR i = 1 TO LEN(OpsDataLine)
        IF MID$(OpsDataLine, i,1) <> ";" THEN
            'still in same field
            FieldValue = FieldValue + MID$(OpsDataLine, i, 1)
        ELSE
            'field changes, so fill content into FlightRec Array
            aOrgDataFileSummer(LineCounter, FieldCounter) = FieldValue
            INCR FieldCounter
            FieldValue = ""
        END IF
    NEXT i

    INCR LineCounter

WEND 'EOF(1)

CLOSE #1

'----- OrgDataFile WINTER -----
CALL LogEntry(FUNCNAME$, "fill winter org data into array")

OPEN FILE_WINTER FOR INPUT AS #1

FILESCAN #1, RECORDS TO TotalLines
DIM aOrgDataFileWinter(1 TO TotalLines+1, 1 TO 36) AS STRING

LineCounter = 1

WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    'parse the OpsDataLine and fill the field variables

    FieldValue = ""
    FieldCounter = 1

    FOR i = 1 TO LEN(OpsDataLine)
        IF MID$(OpsDataLine, i,1) <> ";" THEN
            'still in same field
            FieldValue = FieldValue + MID$(OpsDataLine, i, 1)
        ELSE
            'field changes, so fill content into FlightRec Array
            aOrgDataFileWinter(LineCounter, FieldCounter) = FieldValue
            INCR FieldCounter
            FieldValue = ""
        END IF
    NEXT i

    INCR LineCounter

WEND 'EOF(1)

CLOSE #1

'----- summer -----
CALL LogEntry(FUNCNAME$, "match summer org data and plan data")

FOR OrgDataRec = 1 TO UBOUND(aOrgDataFileSummer, 1)

    'default values for NO MATCH (if there was a match, these values will be overwritten)
    aOrgDataFileSummer(OrgDataRec, 20) = "XXX"           'pax seasonal plan
    aOrgDataFileSummer(OrgDataRec, 14) = "XXX"          'gate seasonal plan
    aOrgDataFileSummer(OrgDataRec, 15) = "XXX"          'pos seasonal plan

    FOR SeasonRec = 1 TO UBOUND(aSummerSeason, 1)

        'do number of weekday and flight number match ?
        'match means: day of week AND flight number

```

```

IF TRIM$(aOrgDataFileSummer(OrgDataRec, 2)) = TRIM$(aSummerSeason(SeasonRec, 1)) AND _
TRIM$(aOrgDataFileSummer(OrgDataRec, 3)) = TRIM$(aSummerSeason(SeasonRec, 2)) THEN

    'yes, they match, so fill OrgDataFileRec with:
    'Number of pax plan / Gate seasonal / pos seasonal
    aOrgDataFileSummer(OrgDataRec, 20) = aSummerSeason(SeasonRec, 5)          'pax seasonal
plan
    aOrgDataFileSummer(OrgDataRec, 14) = aSummerSeason(SeasonRec, 6)         'gate seasonal
plan
    aOrgDataFileSummer(OrgDataRec, 15) = aSummerSeason(SeasonRec, 7)         'pos seasonal
plan

    END IF

NEXT SeasonRec

OrgDataFile
'if there has been no match, it means that there has not been an equivalent of Season record in
'the correspondent OrgDataFile record is written into an error file, indicating the season

IF aOrgDataFileSummer(OrgDataRec, 20) = "XXX" AND _
aOrgDataFileSummer(OrgDataRec, 14) = "XXX" AND _
aOrgDataFileSummer(OrgDataRec, 15) = "XXX" THEN

    'write into ERROR file
    PRINT #2, TRIM$(aOrgDataFileSummer(OrgDataRec, 1)) + ";" + _
              TRIM$(aOrgDataFileSummer(OrgDataRec, 2)) + ";" + _
              TRIM$(aOrgDataFileSummer(OrgDataRec, 3)) + ";SUMMER"

    END IF

NEXT OrgDataRec

'----- winter -----

CALL LogEntry(FUNCNAME$, "match winter org data and plan data")

FOR OrgDataRec = 1 TO UBOUND(aOrgDataFileWinter, 1)

    'default values for NO MATCH (if there was a match, these values will be overwritten)
    aOrgDataFileWinter(OrgDataRec, 20) = "XXX"          'pax seasonal plan
    aOrgDataFileWinter(OrgDataRec, 14) = "XXX"          'gate seasonal plan
    aOrgDataFileWinter(OrgDataRec, 15) = "XXX"          'pos seasonal plan

    FOR SeasonRec = 1 TO UBOUND(aSummerSeason, 1)

        'do number of weekday and flight number match ?
        'match means: day of week AND flight number

        IF TRIM$(aOrgDataFileWinter(OrgDataRec, 2)) = TRIM$(aWinterSeason(SeasonRec, 1)) AND _
TRIM$(aOrgDataFileWinter(OrgDataRec, 3)) = TRIM$(aWinterSeason(SeasonRec, 2)) THEN

            'yes, they match, so fill OrgDataFileRec with:
            'Number of pax plan / Gate seasonal / pos seasonal
            aOrgDataFileWinter(OrgDataRec, 20) = aWinterSeason(SeasonRec, 5)          'pax seasonal
plan
            aOrgDataFileWinter(OrgDataRec, 14) = aWinterSeason(SeasonRec, 6)         'gate seasonal
plan
            aOrgDataFileWinter(OrgDataRec, 15) = aWinterSeason(SeasonRec, 7)         'pos seasonal
plan

            END IF

        NEXT SeasonRec

        OrgDataFile
        'if there has been no match, it means that there has not been an equivalent of Season record in
        'the correspondent OrgDataFile record is written into an error file, indicating the season

        IF aOrgDataFileWinter(OrgDataRec, 20) = "XXX" AND _
aOrgDataFileWinter(OrgDataRec, 14) = "XXX" AND _
aOrgDataFileWinter(OrgDataRec, 15) = "XXX" THEN

            'write into ERROR file
            PRINT #2, TRIM$(aOrgDataFileWinter(OrgDataRec, 1)) + ";" + _
                      TRIM$(aOrgDataFileWinter(OrgDataRec, 2)) + ";" + _
                      TRIM$(aOrgDataFileWinter(OrgDataRec, 3)) + ";WINTER"

            END IF

        NEXT OrgDataRec

        '-----

        'finally write updated OrgDataFile (Summer)

        CALL LogEntry(FUNCNAME$, "generate new summer org data file")

        OPEN LEFT$(FILE_SUMMER, LEN(FILE_SUMMER)-4) + ".new" FOR OUTPUT AS #1

        FOR OrgDataRec = 1 TO UBOUND(aOrgDataFileSummer, 1)

```

```

        CALL WriteRec(aOrgDataFileSummer(), OrgDataRec)
NEXT OrgDataRec

CLOSE #1    'updated OrgDataFile (Summer)
CLOSE #2    'ErrorQueue

'-----

'finally write updated OrgDataFile (Winter)

CALL LogEntry(FUNCNAME$, "generate new winter org data file")

OPEN LEFT$(FILE_WINTER, LEN(FILE_WINTER)-4) + ".new" FOR OUTPUT AS #1

FOR OrgDataRec = 1 TO UBOUND(aOrgDataFileWinter, 1)
    CALL WriteRec(aOrgDataFileWinter(), OrgDataRec)
NEXT OrgDataRec

CLOSE #1    'updated OrgDataFile (Winter)
CLOSE #2    'ErrorQueue

    CALL ShiftFileIntoHistory(LEFT$(FILE_SUMMER, LEN(FILE_SUMMER)-4))
    CALL ShiftFileIntoHistory(LEFT$(FILE_WINTER, LEN(FILE_WINTER)-4))

    CALL LogEntry(FUNCNAME$, "END")
END SUB ' FillPlanningSeason()

'-----

SUB GenerateGanttData()

    CALL LogEntry(FUNCNAME$, "START")

    'allows to select 1 day (on ATD basis),
    'classify retail DATA into a/b/c,
    'write a semicolon delimited file for an import into an EXCEL file with conditional formatting for display

    'select day
    'read day's flight info into array
    'generate empty array for later output file (time,gates), i.e. (290, 153)
    'for each record determine
    ' data: ATD, Standard Ground Time, Gate
    ' determine fields of export matrix to be filled (e.g. 00:10-01:30/Gate A8)
    ' determine category (a/b/c --> color code in EXCEL display)

    '----- DECLARATIONS -----

DIM OpsDataLine AS STRING

DIM TotalLines AS LONG
DIM LineCounter AS LONG

LOCAL FieldValue AS STRING
LOCAL FieldCounter AS INTEGER

LOCAL lCategoryChange AS LONG
DIM SelectedDate AS STRING
DIM NumberOfSelectedRecords AS LONG

DIM SelectedFlightRecs(1 TO %MaxDeparturesPerDay) AS STRING

DIM GanttArray(1 TO 288, 1 TO gNumberOfGates) AS STRING          '(5-min-time slice, gates) e.g. (13:45, B46)
DIM TimeIndex (1 TO 288) AS STRING
LOCAL TimeIndexForATD AS INTEGER
LOCAL TimeIndexForSTD AS INTEGER

    DIM MaxGroundPeriods AS LONG
    LOCAL TaxiOutPeriods AS INTEGER

DIM i AS LONG 'general purpose loop counter
DIM j AS LONG 'general purpose loop counter
DIM z AS LONG 'general purpose loop counter

DIM OutputDay AS INTEGER

DIM FlightRec(1 TO 10, 1 TO 36) AS STRING ' dummy DIM, so that the REDIM works...

DIM lErrorOccurred AS INTEGER

DIM sFieldItems (1 TO 5) AS STRING 'potential fields for output
LOCAL sResult AS STRING
LOCAL nResult AS INTEGER
LOCAL HeatCatLine AS STRING

LOCAL HigherLowerIsBetter AS STRING

'----- ROUTINE -----
ON ERROR GOTO ErrorTrap

```

```

TaxiOutPeriods = 2 'i.e. 2*5 = 10 minutes (used to cope for taxi-out time, as gate is free in the sense of retail
usage)

'----- Select a field for Heat Map view -----

'field 1 is record ID which does not make sense to export

sFieldItems(1) = "Delay Minutes"
sFieldItems(2) = "Flight PAX DF Factor"
sFieldItems(3) = "Retail Revenue (actual)"
sFieldItems(4) = "Retail Revenue (season)"
sFieldItems(5) = "Retail Revenue (opti)"

sResult = ConsoleListBox(20090, _
    LocOfCol(22), _
    LocOfRow(5), _
    "Please select a field for A/B/C Gantt view...", _
    "Field Selection", _
    sFieldItems(), _
    2, _
    %RETURN_INDEX, _
    0)

nResult = VAL(sResult)

SELECT CASE nResult
CASE 1
    nResult = %FIELD_DelayMinutes
    HigherLowerIsBetter = "L"
CASE 2
    nResult = %FIELD_FlightPAXDFfactor
    HigherLowerIsBetter = "H"
CASE 3
    nResult = %FIELD_RetailRevenueActual
    HigherLowerIsBetter = "H"
CASE 4
    nResult = %FIELD_RetailRevenueSeason
    HigherLowerIsBetter = "H"
CASE 5
    nResult = %FIELD_RetailRevenueOpti
    HigherLowerIsBetter = "H"
END SELECT

'----- confirm/change current HEAT MAP CATEGORY BOUNDARIES -----
lCategoryChange = ConsoleMessageBox("Current B-Category is: " + TRIM$(STR$(gHeatCat_B_From)) + "-" +
TRIM$(STR$(gHeatCat_B_To)) + _
    "\n\nWould you like to change [YES] or keep [NO] values?\n", _
    %YESNO+%HANDBOX+%DEFBUTTON1, "INFO", %IDI_QUESTION, %FALSE)

DO
    IF lCategoryChange = %YESBUTTON THEN
        CALL LogEntry(FUNCNAME$, "START: Edit HEATCAT-File")
        SHELL "notepad.exe " + FILE_HEATCAT
        CALL LogEntry(FUNCNAME$, "END: Edit HEATCAT-File")
        OPEN FILE_HEATCAT FOR INPUT AS #1
        WHILE NOT EOF(1)
            LINE INPUT #1, HeatCatLine
            IF LEFT$(HeatCatLine, 2) <> "/" AND LEN(TRIM$(HeatCatLine)) > 0 THEN
                gHeatCat_B_From = VAL(PARSE$(HeatCatLine, ";", 1))
                gHeatCat_B_To = VAL(PARSE$(HeatCatLine, ";", 2))
            END IF
        WEND 'EOF(1)
        CLOSE #1
        IF gHeatCat_B_From < gHeatCat_B_To THEN
            EXIT DO
        ELSE
            ConsoleMessageBox "CHECK: HeatCat B" + STR$(gHeatCat_B_From) + " - " +
STR$(gHeatCat_B_To), %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
            END IF
        ELSE
            EXIT DO
        END IF
    LOOP

'----- select day (ATD is relevant field) to process -----

SelectedDate = "2006-03-26"

SelectedDate = REMOVE$(ConsoleInputBox$(1, %CENTER, %CENTER, _
    "Enter start date of week (Format: YYYY-MM-DD)", _
    "Select Week for Export", SelectedDate, 0, %FALSE), ANY "-")

'----- read all relevant records into an array -----

IF LEN(SelectedDate) = 8 AND NOT ConsoleInputBoxCancel THEN
    'if a date has been entered

    cls

    '----- fill target array with category data -----

```

```

OPEN FILE_TIMEINDEX FOR INPUT AS #1

LineCounter = 1
WHILE NOT EOF(1)
    LINE INPUT #1, TimeIndex(LineCounter)
    INCR LineCounter
WEND 'EOF(1)

CLOSE #1

'ConsoleMessageBox "TimeIndex has been read." & STR$(LineCounter),%DEFAULT,"INFO",%DEFAULT,0

'----- read gate index info into array -----

OPEN FILE_GATE_INFRA FOR INPUT AS #1

FILESCAN #1, RECORDS TO TotalLines
DIM GateArray(1 TO TotalLines+1, 1 TO 7) AS STRING

LineCounter = 1
WHILE NOT EOF(1)
    LINE INPUT #1, OpsDataLine
    IF LEFT$(OpsDataLine, 2) <> "/" THEN                                'bypass comment lines

        'parse the OpsDataLine and fill the field variables

        FieldValue = ""
        FieldCounter = 1

        FOR i = 1 TO LEN(OpsDataLine)
            IF MID$(OpsDataLine, i,1) <> ":" THEN
                'still in same field
                FieldValue = FieldValue + MID$(OpsDataLine, i, 1)
            ELSE
                'field changes, so fill content into Gate Array
                GateArray(LineCounter, FieldCounter) = FieldValue
                INCR FieldCounter
                FieldValue = ""
            END IF
        NEXT i

        INCR LineCounter

    END IF
WEND 'EOF(1)

CLOSE #1

'ConsoleMessageBox "GateIndex has been read." & STR$(LineCounter),%DEFAULT,"INFO",%DEFAULT,0

'----- for each of the day beginning with SelectedDate -----
'-----

FOR OutputDay = 1 TO 7 ' one week of data to be generated

    PRINT "Working on flight data... " + SelectedDate + " (" + TRIM$(STR$(OutputDay)) + " of 7)"

    'delete position in array for use on next day
    FOR i = 1 TO 288 'TimeIndex
        FOR j = 1 TO gNumberOfGates 'GateIndex
            GanttArray(i, j) = ""
        NEXT j
    NEXT i

    '---- try summer season -----

    PRINT "Trying summer season. ";

    LineCounter = 1

    CLOSE #1
    OPEN FILE_SUMMER FOR INPUT AS #1
    FILESCAN #1, RECORDS TO TotalLines
    '--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---
    REDIM FlightRec(1 TO TotalLines+1, 1 TO 36) AS STRING
    '--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---

    WHILE NOT EOF(1)

        LINE INPUT #1, OpsDataLine

        'store only in array if ATD-Date matches the selected date
        IF LEFT$(PARSE$(OpsDataLine, ";", %FIELD_ATD),8) = SelectedDate THEN
            SelectedFlightRecs(LineCounter) = OpsDataLine
            INCR LineCounter
        END IF

    WEND 'EOF(1)

    CLOSE #1

    'try winter season only if there was not a match in summer season (i.e. LineCounter is still 1)

```

```

IF LineCounter = 1 THEN
    '---- try winter season -----
    PRINT " -- NOT FOUND."
    PRINT "Trying winter season.>";
    OPEN FILE_WINTER FOR INPUT AS #1
    FILESCAN #1, RECORDS TO TotalLines
    '--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---
    REDIM FlightRec(1 TO TotalLines+1, 1 TO 36) AS STRING
    '--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---
    WHILE NOT EOF(1)
        LINE INPUT #1, OpsDataLine
        'store only in array if ATD-Date matches the selected date
        IF LEFT$(PARSE$(OpsDataLine, ";", %FIELD_ATD),8) = SelectedDate THEN
            SelectedFlightRecs(LineCounter) = OpsDataLine
            INCR LineCounter
        END IF
    WEND 'EOF(1)
    IF LineCounter > 1 THEN
        PRINT " -- FOUND."
        NumberOfSelectedRecords = LineCounter
    ELSE
        PRINT " -- NOT FOUND."
        NumberOfSelectedRecords = 0
    END IF
    CLOSE #1
    ELSE
        NumberOfSelectedRecords = LineCounter
        PRINT " -- FOUND." 'in SUMMER(!) season
    END IF
    PRINT "Number of Records on that day = " & STR$(NumberOfSelectedRecords)
    PRINT
    'ConsoleMessageBox "Number of Records on that day = " &
    STR$(NumberOfSelectedRecords),%DEFAULT,"INFO",%DEFAULT,0
    '----- generate gantt array -----
    FOR LineCounter = 1 TO NumberOfSelectedRecords
        MaxGroundPeriods = 0
        'check which 5-min. time index the ATD falls into ([001]00:00-[288]23:55)
        'TimeIndexForATD = (hours*12) + INT(minutes/5) + 1
        TimeIndexForATD = VAL(MID$(PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_ATD), 9, 2)) * 12
    'hours
        TimeIndexForATD = TimeIndexForATD + INT(VAL(MID$(PARSE$(SelectedFlightRecs(LineCounter), ";",
    %FIELD_ATD), 11, 2)) / 5)
        TimeIndexForATD = TimeIndexForATD + 1
        'TimeIndexForSTD = (hours*12) + INT(minutes/5) + 1
        TimeIndexForSTD = VAL(MID$(PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_STD), 9, 2)) * 12
    'hours
        TimeIndexForSTD = TimeIndexForSTD + INT(VAL(MID$(PARSE$(SelectedFlightRecs(LineCounter), ";",
    %FIELD_STD), 11, 2)) / 5)
        TimeIndexForSTD = TimeIndexForSTD + 1
        'only for flights with an ATD and with an actual gate
        IF PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_ATD) <> "" AND
        PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_Gate_Actual) <> "" THEN
            'the following cases cope for long delays or early take offs
            'CASE(I) : delay >= standard ground time(SGT) or delay < 0, so FROM= ATD-SGT, TO= ATD-TaxiOutTime
            'CASE(II): delay < standard ground time(SGT), so FROM= STD-SGT, TO= ATD-TaxiOutTime
            'the difference would be for the cases of prior to midnight
            IF VAL(PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_DelayMinutes)) >=
            VAL(PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_StdGroundTime)) OR _
            VAL(PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_DelayMinutes)) < 0 THEN
                'test for a FROM time prior to midnight
                IF INT(VAL(PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_StdGroundTime))/5) >
                TimeIndexForATD THEN
                    'FROM would be previous day (prior to 00:00) so let it start at (FROM) 00:00
                    'calculate the MaxGroundPeriods (5-min-intervals), based on FROM= ATD-SGT, (TO= ATD-
                    TaxiOutTime)
                    MaxGroundPeriods = TimeIndexForATD
                ELSE
                    'FROM is on current day

```

```

TaxiOutTime)          'calculate the MaxGroundPeriods (5-min-intervals), based on FROM= ATD-SGT, (TO= ATD-
MaxGroundPeriods = INT(VAL(PARSE$(SelectedFlightRecs(LineCounter), ";",
%FIELD_StdGroundTime) )/5)
END IF

'fill the GANTTARRAY
FOR j = 0 TO MaxGroundPeriods

'TimeIndexForATD-j ==> go back in time, to cope for standard ground time
IF TimeIndexForATD-TaxiOutPeriods-j > 0 THEN 'fill the target array with the
value of target field (nResult)
GanttArray(TimeIndexForATD-TaxiOutPeriods-j,
GetGateColumn(PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_Gate_Actual), GateArray() ) ) = _
GetCategoryABC(PARSE$(SelectedFlightRecs(LineCounter), ";", nResult), HigherLowerIsBetter)
END IF

NEXT j

ELSE 'CASE(II)
'delay < SGT
'test for a FROM time prior to midnight
IF INT(VAL(PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_StdGroundTime))/5) >
TimeIndexForSTD THEN
TaxiOutTime)          'FROM would be previous day (prior to 00:00) so let it start at (FROM) 00:00
'calculate the MaxGroundPeriods (5-min-intervals), based on FROM= STD-SGT, (TO= ATD-
TaxiOutTime)          MaxGroundPeriods = TimeIndexForSTD
ELSE
'FROM is on current day
'calculate the MaxGroundPeriods (5-min-intervals), based on FROM= ATD-SGT, (TO= ATD-
TaxiOutTime)          MaxGroundPeriods = INT(VAL(PARSE$(SelectedFlightRecs(LineCounter), ";",
%FIELD_StdGroundTime))/5)
END IF

'fill the GANTTARRAY
FOR j = 0 TO MaxGroundPeriods

'TimeIndexForSTD-j ==> go back in time, to cope for standard ground time
IF TimeIndexForSTD-TaxiOutPeriods-j > 0 THEN 'fill the target array with the
value of target field (nResult)
GanttArray(TimeIndexForSTD-TaxiOutPeriods-j, GetGateColumn(
PARSE$(SelectedFlightRecs(LineCounter), ";", %FIELD_Gate_Actual), GateArray() ) ) = _
GetCategoryABC(PARSE$(SelectedFlightRecs(LineCounter), ";", nResult), HigherLowerIsBetter)
END IF

NEXT j

END IF 'CASE I and II

END IF 'only flights with ATD and with actual gate

NEXT LineCounter

'ConsoleMessageBox "Gantt Array has been generated. ",%DEFAULT,"INFO",%DEFAULT,0

'----- write output file for GANTT view in EXCEL -----

IF NumberOfSelectedRecords > 0 THEN

OPEN FILE_GANTTVIEW + "_" + SelectedDate + ".TXT" FOR OUTPUT AS #2
PRINT #2, "// Field      : " + sFieldItems(VAL(sResult))
PRINT #2, "// Category B: " + TRIM$(STR$(gHeatCat_B_From)) + "-" + TRIM$(STR$(gHeatCat_B_To))
PRINT #2, "//"

FOR i = 1 TO 288 'TimeIndex

OpsDataLine = ""
FOR j = 1 TO gNumberOfGates 'GateIndex
OpsDataLine = OpsDataLine + GanttArray(i, j) + ";"
'delete position in array for use on next day
GanttArray(i, j) = ""
NEXT j 'GateIndex

PRINT #2, OpsDataLine

NEXT i 'TimeIndex

CLOSE #2

END IF

SelectedDate = NextDay(SelectedDate)

NEXT OutputDay

ConsoleMessageBox "If dates have been found, you may import the data into a pre-formatted EXCEL sheet\n\nor
use the Animate Data Function.",%DEFAULT,"INFO",%DEFAULT,0

```



```

cls
PRINT "Cleaning up..."

ELSE

    ConsoleMessageBox "Nothing has been exported. ",%DEFAULT,"INFO",%DEFAULT,0

END IF 'LEN(SelectedDate) = 8

CALL LogEntry(FUNCNAME$, "END")

GENERATEGANTTDATA_RESUME:
IF lErrorOccurred = %TRUE THEN
    ConsoleMessageBox "Terminating Procedure.",%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
cls
    EXIT SUB
END IF

EXIT SUB

ErrorTrap:

lErrorOccurred = %TRUE
ConsoleMessageBox "ERROR OCCURRED: " + STR$(ERR) + " " +
ERROR$,%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

RESUME GENERATEGANTTDATA_RESUME

END SUB 'GenerateGanttData()

'-----

FUNCTION GetGateColumn (BYVAL Gate AS STRING, BYREF GateArray() AS STRING) AS INTEGER

    DIM i AS LONG

    GetGateColumn = 0

    IF Gate <> "" THEN
        FOR i = 1 TO UBOUND(GateArray, 1)

            IF TRIM$(GateArray(i,2)) = TRIM$(Gate) THEN
                GetGateColumn = VAL(GateArray(i,1))
                EXIT FUNCTION
            END IF

        NEXT i
    END IF

END FUNCTION 'GetGateColumn()

'-----

FUNCTION GetCategoryABC(BYVAL FieldValue AS STRING, BYVAL ValueDirection AS STRING) AS STRING

    REPLACE ", " WITH "." IN ValueDirection 'just in case...

    IF ValueDirection = "H" THEN ' higher is better

        SELECT CASE VAL(FieldValue)
            CASE < gHeatCat_B_From ' approx. 1st Quartile
                GetCategoryABC = "c"
            CASE gHeatCat_B_From TO gHeatCat_B_To ' approx. 2nd and >3rd Quartile
                GetCategoryABC = "b"
            CASE > gHeatCat_B_To ' approx. top 20% (80% coverage)
                GetCategoryABC = "a"
            CASE ELSE
                GetCategoryABC = "x"
        END SELECT

    ELSE ' lower is better

        SELECT CASE VAL(FieldValue)
            CASE < gHeatCat_B_From ' approx. 1st Quartile
                GetCategoryABC = "a"
            CASE gHeatCat_B_From TO gHeatCat_B_To ' approx. 2nd and >3rd Quartile
                GetCategoryABC = "b"
            CASE > gHeatCat_B_To ' approx. top 20% (80% coverage)
                GetCategoryABC = "c"
            CASE ELSE
                GetCategoryABC = "x"
        END SELECT

    END IF

END FUNCTION 'GetCategoryABC()

```



```

aPax( DayOfWeekInSeason, aElementCounter(DayOfWeekInSeason) ) =
PARSE$(OpsDataLine, ";", %FIELD_PAX_Actual)
ELSE 'if there is no valid actual pax figure take the booked pax figure (if
there is one)
IF VAL (PARSE$(OpsDataLine, ";", %FIELD_PAX_Booked)) <> 0 THEN
aPax( DayOfWeekInSeason,
aElementCounter(DayOfWeekInSeason) ) = PARSE$(OpsDataLine, ";", %FIELD_PAX_Booked)
ELSE
'if no booked pax either try on plan values
IF VAL (PARSE$(OpsDataLine, ";", %FIELD_PAX_Season)) <> 0
THEN
aPax( DayOfWeekInSeason,
aElementCounter(DayOfWeekInSeason) ) = PARSE$(OpsDataLine, ";", %FIELD_PAX_Season)
ELSE
'look in FLIRT data
lFLIRTpax = 0
FOR curFLIRTRec = 1 TO UBOUND(aFLIRT)
IF PARSE$(aFLIRT(curFLIRTRec), ";", 2)
= PARSE$(OpsDataLine, ";", %FIELD_FlightNumber) THEN
aPax( DayOfWeekInSeason,
aElementCounter(DayOfWeekInSeason) ) = PARSE$(aFLIRT(curFLIRTRec), ";", %FIELD_FlightNumber)
lFLIRTpax = 1
END IF
NEXT curFLIRTRec
IF lFLIRTpax = 0 THEN
aPax( DayOfWeekInSeason,
aElementCounter(DayOfWeekInSeason) ) = "xxx"
PRINT #3, OpsDataLine
END IF
END IF
END IF
END IF
INCR aElementCounter(DayOfWeekInSeason)
END IF
WEND 'EOF(1)
CLOSE #1
NEXT i
FOR i = 1 TO %MaxDeparturesPerDay * (INT(183/7)+1)
OpsDataLine = ""
FOR j = 1 TO 14
OpsDataLine = OpsDataLine + aPax(j,i) + ";"
NEXT j
'delete the last delimiter
OpsDataLine = LEFT$(OpsDataLine, LEN(OpsDataLine)-1)
PRINT #2, OpsDataLine
NEXT i
CLOSE #2
CLOSE #3
ConsoleMessageBox "PAX file generated. For any problems look into file: \n" + PATH_APPLICATION+FILE_DEBUG,
%DEFAULT,"INFO",%DEFAULT,0
CALL LogEntry(FUNCNAME$, "END")
END SUB 'GenerateWeekDayPAXFile()
'-----
SUB GenerateWeekDayFIELDFile()
CALL LogEntry(FUNCNAME$, "START")
'generates a file to be imported by EXCEL for further stats
'content are actual field/attribute figures per flight grouped by weekday across the according season (both
seasons)
'form: DAY1;DAY2;DAY3;DAY4;DAY5;DAY6;DAY7;DAY1;DAY2;DAY3;DAY4;DAY5;DAY6;DAY7
' 223;333;123;120;333;87;97;...
' 301;97;101;224;322;54;124
' ...
'----- DECLARATIONS -----
DIM aSeason(1 TO 2) AS STRING
LOCAL i AS LONG
LOCAL j AS LONG
LOCAL ExportField AS INTEGER

```

```

DIM aElementCounter(1 TO %MaxDeparturesPerDay * (INT(183/7)+1) ) AS LONG '183
days = approx. one season

'each of
the 7 days has %MaxDeparturesPerDay
LOCAL DayOfWeekInSeason AS INTEGER
LOCAL OpsDataLine AS STRING

LOCAL DateTimeStamp AS STRING
LOCAL lFilter AS LONG

DIM aExportField(1 TO 14, 1 TO %MaxDeparturesPerDay * (INT(183/7)+1) ) AS STRING 'target array
(flight event ; day 1..7 = summer / day 8..14 = winter)

LOCAL LineCounter AS LONG
LOCAL DismissedLines AS LONG
LOCAL PaxCount AS LONG

'----- ROUTINE -----

lFilter = %YESBUTTON

aSeason(1) = FILE_SUMMER
aSeason(2) = FILE_WINTER

LineCounter = 0
DismissedLines = 0
PaxCount = 0

'initialize the array
FOR i = 1 TO 14
  FOR j = 1 TO %MaxDeparturesPerDay * (INT(183/7)+1)
    aExportField(i,j) = "---"
  NEXT j
NEXT i

'select the field to export
ExportField = SelectField()

IF ExportField >1 AND ExportField <37 THEN

  cls
  PRINT "selected Field = " + TRIM$(STR$(ExportField))

  'for comparison purpose of retail revenue ACTUAL revenue can
  'only compared to e.g. SEASONAL PLANNING revenue when just
  'those records are exported that have both ACTUAL and SEASON revenue filled

  IF ExportField = %FIELD_RetailRevenueActual THEN
    lFilter = ConsoleMessageBox("Do you want to export all records (YES) or just those\n\n" + _
      "that also have SEASONAL revenue filled (NO) ? ", _
      %YESNO+%HANDBOX+%DEFBUTTON1, "INFO", %IDI_QUESTION, %FALSE)
  END IF

  IF ExportField = %FIELD_DelayMinutes THEN
    PRINT "Any entry of '9999' will be set to '0'."
  END IF

  IF lFilter = %YESBUTTON THEN
    PRINT "All records will be exported."
  ELSE
    PRINT "Only records with both ACTUAL and SEASONAL revenue will be exported."
  END IF

  OPEN PATH_APPLICATION+$FILE_DEBUG FOR APPEND AS #3
  PRINT #3, "NEW ENTRY: " + DATE$ + " / " + TIME$

  DateTimeStamp = TIME$
  REPLACE ANY ":" WITH "-" IN DateTimeStamp
  DateTimeStamp = DATE$ + "_" + DateTimeStamp
  OPEN PATH_DATA + $FILE_OUTPUT + "_Field_" + TRIM$(STR$(ExportField)) + "_" + DateTimeStamp + ".txt" FOR OUTPUT
AS #2

  'for each of the seasons read the actual pax data
  PRINT "Reading field values to be exported..."

  FOR i = 1 TO UBOUND(aSeason)

    ARRAY ASSIGN aElementCounter() = 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

    OPEN aSeason(i) FOR INPUT AS #1

    WHILE NOT EOF(1)

      INCR LineCounter
      LINE INPUT #1, OpsDataLine

      'only for flights with an ATD
      IF PARSE$(OpsDataLine, ";", %FIELD_ATD) <> "" THEN
        IF lFilter = %YESBUTTON THEN

          DayOfWeekInSeason = VAL( PARSE$(OpsDataLine, ";", %FIELD_DayOfWeek) ) + 7*(i-

```

```

IF ExportField = %FIELD_DelayMinutes AND PARSE$(OpsDataLine, ";", ExportField)
= "9999" THEN
    'set value to zero
    aExportField( DayOfWeekInSeason, aElementCounter(DayOfWeekInSeason) ) =
"0"
ELSE
    aExportField( DayOfWeekInSeason, aElementCounter(DayOfWeekInSeason) ) =
PARSE$(OpsDataLine, ";", ExportField)
END IF

PaxCount = PaxCount + VAL(PARSE$(OpsDataLine, ";", %FIELD_PAX_Actual) )
INCR aElementCounter(DayOfWeekInSeason)

ELSE 'apply filter for field 34: %FIELD_RetailRevenueActual depending on field 35:
%FIELD_RetailRevenueSeason
'--> export only if there is both ACTUAL and SEASONAL revenue
IF VAL(PARSE$(OpsDataLine, ";", %FIELD_RetailRevenueSeason)) <> 0 THEN
    DayOfWeekInSeason = VAL( PARSE$(OpsDataLine, ";", %FIELD_DayOfWeek) ) + 7*(i-1)
    aExportField( DayOfWeekInSeason, aElementCounter(DayOfWeekInSeason)
) = PARSE$(OpsDataLine, ";", ExportField)
    PaxCount = PaxCount + VAL(PARSE$(OpsDataLine, ";",
%FIELD_PAX_Actual) )
    INCR aElementCounter(DayOfWeekInSeason)
ELSE
    INCR DismissedLines
END IF
END IF
WEND 'EOF(1)
CLOSE #1

NEXT i

PRINT "Writing output file..."

FOR i = 1 TO %MaxDeparturesPerDay * (INT(183/7)+1)

    OpsDataLine = ""
    FOR j = 1 TO 14
        OpsDataLine = OpsDataLine + aExportField(j,i) + ";"
    NEXT j

    'delete the last delimiter
    OpsDataLine = LEFT$(OpsDataLine, LEN(OpsDataLine)-1)

    PRINT #2, OpsDataLine

NEXT i

PRINT #3, "Total Recs with ATD: " + STR$(LineCounter) + _
" / Dismissed Lines (in case of ACTUAL REVENUE FILTER): " + STR$(DismissedLines) + _
" / PaxCount: " + STR$(PaxCount)

CLOSE #2
CLOSE #3

ConsoleMessageBox "Outfile file generated.\nBut remember ONLY flights with an ATD have been
considered.\n\n" + _
"For any problems look into file: \n" + PATH_APPLICATION+$FILE_DEBUG,
%DEFAULT,"INFO",%DEFAULT,0

ELSE
    ConsoleMessageBox "No field chosen. Nothing exported.", %DEFAULT,"INFO",%DEFAULT,0
END IF

cls

CALL LogEntry(FUNCNAME$, "END: Field = " + TRIM$(STR$(ExportField)) )

END SUB 'GenerateWeekDayFIELDFile()

'-----

FUNCTION SelectField() AS INTEGER

DIM sFieldItems (2 TO 36) AS STRING 'Number of fields in data file
LOCAL sResult AS STRING

SelectField = 0

'field 1 is record ID which does not make sense to export

sFieldItems(2) = "Day Of Week"
sFieldItems(3) = "Flight Number"
sFieldItems(4) = "STD"
sFieldItems(5) = "ETD"
sFieldItems(6) = "ATD"

```

```

sFieldItems(7) = "Delay Minutes"
sFieldItems(8) = "Aircraft Type"
sFieldItems(9) = "Wing Span Code"
sFieldItems(10) = "CKI Hall"
sFieldItems(11) = "Terminal Building"
sFieldItems(12) = "Gate (actual)"
sFieldItems(13) = "Stand (actual)"
sFieldItems(14) = "Gate (season)"
sFieldItems(15) = "Stand (season)"
sFieldItems(16) = "Gate (Opti)"
sFieldItems(17) = "Stand (Opti)"
sFieldItems(18) = "[PAX (booked)]"
sFieldItems(19) = "[PAX (actual)]"
sFieldItems(20) = "[PAX (season)]"
sFieldItems(21) = "SLF (actual)"
sFieldItems(22) = "Gate Size (actual)"
sFieldItems(23) = "Gate Size (season)"
sFieldItems(24) = "Gate Size (opti)"
sFieldItems(25) = "Standard Ground Time"
sFieldItems(26) = "Flight PAX DF Factor"
sFieldItems(27) = "Destination Country"
sFieldItems(28) = "Retail Area (actual)"
sFieldItems(29) = "Retail Area Factor (actual)"
sFieldItems(30) = "Retail Area (season)"
sFieldItems(31) = "Retail Area Factor (season)"
sFieldItems(32) = "Retail Area (opti)"
sFieldItems(33) = "Retail Area Factor (opti)"
sFieldItems(34) = "Retail Revenue (actual)"
sFieldItems(35) = "Retail Revenue (season)"
sFieldItems(36) = "Retail Revenue (opti)"

sResult = ConsoleListBox(200100, _
    LocOfCol(22), _
    LocOfRow(5), _
    "Please select a field to export ...", _
    "Field Export", _
    sFieldItems(), _
    19, _
    %RETURN_INDEX, _
    0)

SelectField = VAL(sResult)+1

' ConsoleMessageBox _
'     "You selected\n\n"+_
'     sResult+"\n\n"+_
'     "which is: " + sFieldItems(VAL(sResult)+1), _
'     %OKONLY,"",%DEFAULT,0
'

END FUNCTION 'SelectField()

'-----

SUB DeleteRecords(BYVAL FileName AS STRING, BYVAL FieldNumber AS INTEGER, BYVAL MatchingValue AS STRING)

    CALL LogEntry(FUNCNAME$, "START: " + FileName + " Field: " + STR$(FieldNumber) + "Value: " + MatchingValue)

    'generic procedure
    'used to filter out (delete) records that are not of relevance to further research
    'MatchingValue is the value upon which a record is deleted/not taken over into new file

    LOCAL OpsDataLine AS STRING
    LOCAL NumberOfDeletedRecords AS LONG
    LOCAL NewFileName AS STRING

    NumberOfDeletedRecords = 0

    NewFileName = LEFT$(FileName, LEN(FileName)-4) + ".new"

    OPEN FileName FOR INPUT AS #1
    OPEN NewFileName FOR OUTPUT AS #2

    WHILE NOT EOF(1)

        LINE INPUT #1, OpsDataLine

        'if no match then write in new file, else omit record
        IF PARSE$(OpsDataLine, ";", FieldNumber) <> MatchingValue THEN
            PRINT #2, OpsDataLine
        ELSE
            INCR NumberOfDeletedRecords
        END IF

    WEND 'EOF(1)

    ConsoleMessageBox "Number of deleted records: " + STR$(NumberOfDeletedRecords), %DEFAULT,"INFO",%DEFAULT,0

    CLOSE #1
    CLOSE #2

```

```

CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))
CALL LogEntry(FUNCNAME$, "END: " + FileName + " Field: " + STR$(FieldNumber) + "Value: "+ MatchingValue)
END SUB 'DeleteRecords()

'-----

SUB DeleteRecordsTWO(BYVAL FileName AS STRING)
CALL LogEntry(FUNCNAME$, "START: " + FileName)
'delete records with no ATD AND NO ACTUAL PAX
LOCAL OpsDataLine AS STRING
LOCAL NumberOfDeletedRecords AS LONG
LOCAL NewFileName AS STRING

NumberOfDeletedRecords = 0

'used to filter out (delete) records that are not of relevance to further research
NewFileName = LEFT$(FileName, LEN(FileName)-4) + ".new"

OPEN FileName FOR INPUT AS #1
OPEN NewFileName FOR OUTPUT AS #2
OPEN LEFT$(NewFileName, LEN(NewFileName)-4) + ".DEL" FOR OUTPUT AS #3

WHILE NOT EOF(1)
    LINE INPUT #1, OpsDataLine
    'if no ATD and no ActualPax
    IF PARSE$(OpsDataLine, ";", %FIELD_ATD) = "" AND VAL(PARSE$(OpsDataLine, ";", %FIELD_PAX_Actual))
= 0 THEN
        PRINT #3, OpsDataLine
        INCR NumberOfDeletedRecords
    ELSE
        'either there is an ATD or there is a figure for ActualPax, so write into target file
        PRINT #2, OpsDataLine
    END IF
WEND 'EOF(1)

ConsoleMessageBox "Number of deleted records: " + STR$(NumberOfDeletedRecords), %DEFAULT, "INFO", %DEFAULT, 0

CLOSE #1
CLOSE #2
CLOSE #3

CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))
CALL LogEntry(FUNCNAME$, "END: " + FileName)
END SUB 'DeleteRecordsTWO()

'-----

SUB FillAssumedATD(BYVAL FileName AS STRING)
CALL LogEntry(FUNCNAME$, "START: " + FileName)
'for those flights that have no ATD but actual pax figures, take (if existent) STD or better ETD as its ATD
'flight with no pax are deleted
DIM OpsDataLine AS STRING
DIM NumberOfUpdatedRecords AS LONG
DIM NewFileName AS STRING

DIM Field_STD AS INTEGER
DIM Field_ATD AS INTEGER
DIM Field_ETD AS INTEGER

DIM NoPaxCounter AS LONG

NumberOfUpdatedRecords = 0
NoPaxCounter = 0

'used to filter out (delete) records that are not of relevance to further research
NewFileName = LEFT$(FileName, LEN(FileName)-4) + ".new"

OPEN FileName FOR INPUT AS #1
OPEN NewFileName FOR OUTPUT AS #2
OPEN LEFT$(NewFileName, LEN(NewFileName)-4) + ".ATDUPD" FOR OUTPUT AS #3

WHILE NOT EOF(1)

```

```

LINE INPUT #1, OpsDataLine

'if no ATD but Acutal Pax
IF PARSE$(OpsDataLine, ";", %FIELD_ATD) = "" AND VAL(PARSE$(OpsDataLine, ";", %FIELD_PAX_Actual))
<> 0 THEN

    'check for ETD
    IF PARSE$(OpsDataLine, ";", %FIELD_ETD) <> "" THEN

        'update ATD with ETD info
        OpsDataLine = StringUpdate(OpsDataLine, %FIELD_ATD, PARSE$(OpsDataLine, ";",
%FIELD_ETD))

        PRINT #3, "ETD | " + OpsDataLine
        PRINT #2, OpsDataLine
        INCR NumberOfUpdatedRecords

    ELSE

        'no ETD, so check for STD
        IF PARSE$(OpsDataLine, ";", %FIELD_STD) <> "" THEN
            'update ATD with STD info
            OpsDataLine = StringUpdate(OpsDataLine, %FIELD_ATD,
PARSE$(OpsDataLine, ";", %FIELD_STD))

            PRINT #3, "STD | " + OpsDataLine
            PRINT #2, OpsDataLine
            INCR NumberOfUpdatedRecords

        END IF

    END IF

ELSE

    'either there is an ATD or there is a figure for ActualPax, so write into target file
    'so there might be flights with no actual pax, but an ATD
    'write back recs with valid pax figure only (pax>0), i.e. delete recs with no actual pax

    IF VAL(PARSE$(OpsDataLine, ";", %FIELD_PAX_Actual)) = 0 THEN
        INCR NoPaxCounter
        PRINT #3, "0PAX| " + OpsDataLine
    ELSE
        PRINT #2, OpsDataLine
    END IF

END IF

WEND 'EOF(1)

ConsoleMessageBox "Updated records: " + STR$(NumberOfUpdatedRecords) + "\nDeleted Recs with no PAX: "+
STR$(NoPaxCounter), %DEFAULT, "INFO", %DEFAULT, 0

CLOSE #1
CLOSE #2
CLOSE #3

CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))

CALL LogEntry(FUNCNAME$, "END: " + FileName)

END SUB 'FillAssumedATD()

'-----

SUB GenerateTable_AvgOnFieldUnique(BYVAL FileName AS STRING, BYVAL MatchFieldNumber AS INTEGER, BYVAL
ComputeFieldNumber AS INTEGER)

    CALL LogEntry(FUNCNAME$, "START")

    'generates a table (file) with unique flight numbers and average pax actual on that flight number
    'MatchFieldNumber:      determines field that forms the unique list (e.g. flight no, a/c type, gate,
...)
    'ComputeFieldNumber:    determines field which values will be taken to calculate the average

    '----- DECLARATIONS -----

    DIM OpsDataLine AS STRING
    DIM IsInArray AS INTEGER
    DIM ArrayPointer AS LONG
    DIM TotalLines AS LONG
    LOCAL NewFileName AS STRING

    DIM i AS LONG

    '----- ROUTINE -----

    OPEN FileName FOR INPUT AS #1

```



```

NewFileName = LEFT$(FileName, LEN(FileName)-4) + "_UNIQUE_FIELD_" + TRIM$(STR$(MatchFieldNumber)) +
"_AVG_FIELD_" + TRIM$(STR$(ComputeFieldNumber)) + ".txt"
OPEN NewFileName FOR OUTPUT AS #2

FILESCAN #1, RECORDS TO TotalLines
DIM UniqueEntry(1 TO TotalLines+1, 1 TO 3) AS STRING

FOR i = 1 TO TotalLines+1
    UniqueEntry(i,1) = "---"
    UniqueEntry(i,2) = "0"
    UniqueEntry(i,3) = "0"
NEXT i

ArrayPointer = 0

WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    IsInArray = 0

    'if not yet in array
    FOR i = 1 TO ArrayPointer 'test each element in array

        IF UniqueEntry(i,1) = PARSE$(OpsDataLine, ";", MatchFieldNumber) THEN
            'is already in array
            'insert into array if there is a valid numerical value not zero

                IF PARSE$(OpsDataLine, ";", ComputeFieldNumber) <> "" AND
VAL(PARSE$(OpsDataLine, ";", ComputeFieldNumber)) <> 0 THEN

                    UniqueEntry(i,2) = STR$(VAL(UniqueEntry(i,2)) + VAL(PARSE$(OpsDataLine, ";",
ComputeFieldNumber)))
                    'Add field value to current sum
                    UniqueEntry(i,3) = STR$(VAL(UniqueEntry(i,3)) + 1)
                    'increase

ocurance by 1

                    i = ArrayPointer

                    'to exit the loop
                    IsInArray = 1

                END IF

            END IF

        NEXT i

    IF IsInArray = 0 THEN ' if not yet in array

        'insert into array if there is a valid numerical value not zero
        IF PARSE$(OpsDataLine, ";", ComputeFieldNumber) <> "" AND VAL(PARSE$(OpsDataLine, ";",
ComputeFieldNumber)) <> 0 THEN

            INCR ArrayPointer
            UniqueEntry(ArrayPointer,1)= PARSE$(OpsDataLine, ";", MatchFieldNumber)
            ' Content of MatchField
            UniqueEntry(ArrayPointer,2)= STR$(VAL(PARSE$(OpsDataLine, ";", ComputeFieldNumber)))
            ' Content of ComputeField
            UniqueEntry(ArrayPointer,3)= "1"
            ' First occurrence

        END IF

    END IF ' if not yet in array

WEND 'EOF(1)

FOR i = 1 TO ArrayPointer-1
    IF VAL(UniqueEntry(i,3)) <> 0 AND VAL(UniqueEntry(i,2)) <> 0 THEN
        PRINT #2, UniqueEntry(i,1) + ";" + TRIM$(STR$( ROUND( (VAL(UniqueEntry(i,2)) /
VAL(UniqueEntry(i,3)) ), 0) ))
    ELSE

        ConsoleMessageBox "ZERO VALUES: " + UniqueEntry(i,2) + " / " + UniqueEntry(i,3), %DEFAULT,"INFO",%DEFAULT,0

    END IF
NEXT i

CLOSE #1
CLOSE #2

CALL LogEntry(FUNCNAME$, "END")
END SUB 'GenerateTable_AvgOnFieldUnique()

'-----

SUB FillActualPax(BYVAL FileName AS STRING)

    CALL LogEntry(FUNCNAME$, "START")

```

```

'tries to find a pax figure in the help tables and updates the file with it

'----- DECLARATIONS -----

LOCAL NewFileName AS STRING
LOCAL TotalLines AS LONG

LOCAL i AS LONG
LOCAL LineCounter AS LONG
LOCAL curFLIRTRec AS LONG
LOCAL OpsDataLine AS STRING
LOCAL lUpdate AS INTEGER
LOCAL lFLIRTpax AS INTEGER

'----- ROUTINE -----

'read flight numbers with avg actual pax into an array

NewFileName = LEFT$(FileName, LEN(FileName)-4) + "_UNIQUE_FIELD_3_AVG_FIELD_19.txt"

'read FLIRT data into array in case no pax figure is found
OPEN FILE_FLIRT FOR INPUT AS #1

    FILESCAN #1, RECORDS TO i
    DIM aFLIRT(1 TO i) AS STRING
    CLOSE #1

OPEN FILE_FLIRT FOR INPUT AS #1
i = 1
WHILE NOT EOF(1)
    LINE INPUT #1, aFLIRT(i)
    INCR i
WEND 'EOF(1)
CLOSE #1

OPEN NewFileName FOR INPUT AS #1
FILESCAN #1, RECORDS TO TotalLines
CLOSE #1

OPEN NewFileName FOR INPUT AS #1

DIM FlightNumber(1 TO TotalLines) AS STRING

LineCounter = 1
WHILE NOT EOF(1)
    LINE INPUT #1, FlightNumber(LineCounter)
    INCR LineCounter
WEND 'EOF(1)
CLOSE #1

'read flight records and check for missing actual pax value
'if values missing then get from array

OPEN FileName FOR INPUT AS #1
OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #2
OPEN LEFT$(FileName, LEN(FileName)-4) + ".ERR" FOR OUTPUT AS #3

WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine
    IF VAL(PARSE$(OpsDataLine, ";", %FIELD_PAX_Actual)) = 0 THEN
        'get from array
        lUpdate = 0
        FOR i = 1 TO UBOUND(FlightNumber)
            IF PARSE$(OpsDataLine, ";", %FIELD_FlightNumber) = PARSE$(FlightNumber(i),
";", 1) THEN
                lUpdate = 1
                'update that record
                OpsDataLine = StringUpdate(OpsDataLine, %FIELD_PAX_Actual,
PARSE$(FlightNumber(i), ";", 2))
            END IF
        NEXT i

        'if no match in array has been found then print into error file
        IF lUpdate = 0 THEN

            'look in FLIRT data (for a last chance)
            lFLIRTpax = 0
            FOR curFLIRTRec = 1 TO UBOUND(aFLIRT)
                IF PARSE$(aFLIRT(curFLIRTRec), ";", 2) = PARSE$(OpsDataLine, ";", 3)
THEN
                    OpsDataLine = StringUpdate(OpsDataLine, %FIELD_PAX_Actual,
PARSE$(aFLIRT(curFLIRTRec), ";", 3))
                END IF
            NEXT curFLIRTRec

            IF lFLIRTpax = 0 THEN
                PRINT #3, OpsDataLine
            END IF
        END IF
    END IF
END IF

```

```

                                END IF

                                END IF

                                'write that record
                                PRINT #2, OpsDataLine

                                WEND 'EOF(1)

                                CLOSE #1
                                CLOSE #2
                                CLOSE #3

                                CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))

                                ConsoleMessageBox "For the following records no PAX have been found: ", %DEFAULT,"INFO",%DEFAULT,0

                                SHELL "notepad.exe " + LEFT$(FileName, LEN(FileName)-4) + ".ERR"

                                CALL LogEntry(FUNCNAME$, "END")
END SUB 'FillActualPax()

'-----

SUB Conversion()

    CALL LogEntry(FUNCNAME$, "START")

    'filters relevant records with relevant fields from FLIRT file

    LOCAL OpsDataLine AS STRING

    OPEN FILE_FLIRTOrg FOR INPUT AS #1
    OPEN FILE_FLIRT FOR OUTPUT AS #2

    WHILE NOT EOF(1)
        LINE INPUT #1, OpsDataLine
        'only passenger flights (no cargo) and only relevant fields
        IF PARSE$(OpsDataLine, ";", 2) = "S" AND (PARSE$(OpsDataLine, ";", 10) = "PP" OR PARSE$(OpsDataLine,
";", 10) = "SP" OR PARSE$(OpsDataLine, ";", 10) = "PC") THEN
            OpsDataLine = PARSE$(OpsDataLine, ";", 3) + ";" + PARSE$(OpsDataLine, ";", 5) +
SPACE$(3-LEN(PARSE$(OpsDataLine, ";", 5))) + FORMAT$(VAL(PARSE$(OpsDataLine, ";", 6)), "00000") + ";" +
PARSE$(OpsDataLine, ";", 14)
            PRINT #2, OpsDataLine
        END IF
    WEND

    CLOSE #1
    CLOSE #2

    CALL LogEntry(FUNCNAME$, "END")
END SUB 'Conversion()

'-----

SUB ShiftFileIntoHistory(BYVAL FileName AS STRING)

    CALL LogEntry(FUNCNAME$, "START: " + FileName)

    ' FileName is the filename (incl. path) WITHOUT extension

    'if within a routine, a data file has been updated, it has gotten the extension ".new" (in that routine)
    'the original file has the extension ".txt"
    'this procedure copies the .txt-file into a history directory with a date/time-stamp at the end of its file
name
    'the .new-file becomes the .txt-file

    LOCAL FileNameCore AS STRING
    LOCAL DateTimeStamp AS STRING

    DateTimeStamp = TIME$
    REPLACE ANY ":" WITH "-" IN DateTimeStamp
    DateTimeStamp = DATE$ + "_" + DateTimeStamp

    FileNameCore = RIGHT$(FileName, LEN(FileName)-LEN(PATH_DATA))

    FILECOPY FileName + ".txt", PATH_HISTORY + FileNameCore + "_" + DateTimeStamp + ".txt"
    SLEEP 1000
    KILL FileName + ".txt"
    NAME FileName + ".new" AS FileName + ".txt"

    CALL LogEntry(FUNCNAME$, "END: " + FileName)
END SUB 'ShiftFileIntoHistory()

'-----

```

```

SUB SuggestRetailAreaFactor(BYVAL FileName AS STRING)

'suggests a retail area factor that can be entered / edited into the according data file
'based on: avg. values for FlightPAXDFfactor

CALL LogEntry(FUNCNAME$, "START")

LOCAL OpsDataLine AS STRING
LOCAL LineCounter AS LONG
LOCAL TotalLines AS LONG
LOCAL NumberOfRetailAreas AS INTEGER
LOCAL TempFactor AS STRING
LOCAL i AS LONG
LOCAL lResult AS LONG
LOCAL MinValue AS DOUBLE
LOCAL NO_RA_Counter AS LONG

'-----

'get a suggestion

'count number of different defined retail areas

OPEN FILE_RETAIL_AREA_DEF FOR INPUT AS #1
LineCounter = 0
WHILE NOT EOF(1)
  LINE INPUT #1, OpsDataLine
  IF LEFT$(OpsDataLine,2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN 'no comment line in data file
    INCR LineCounter
  END IF
WEND 'EOF(1)
CLOSE #1
NumberOfRetailAreas = LineCounter

'now DIM the array and fill with Retail Areas Names

DIM aRetailArea (1 TO NumberOfRetailAreas, 1 TO 4) AS STRING

OPEN FILE_RETAIL_AREA_DEF FOR INPUT AS #1
LineCounter = 1
WHILE NOT EOF(1)
  LINE INPUT #1, OpsDataLine
  IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file
    aRetailArea(LineCounter, 1) = TRIM$(PARSE$(OpsDataLine, ";", 1))
    aRetailArea(LineCounter, 2) = "0"
    aRetailArea(LineCounter, 3) = "0"
    INCR LineCounter
  END IF
WEND 'EOF(1)
CLOSE #1

'work on datafile now
OPEN FileName FOR INPUT AS #1
FILESCAN #1, RECORDS TO TotalLines 'used for progress calculation

ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing "+TRIM$(USING$(###,###,###),
TotalLines))+ " Records.", "Reading file...", %FALSE

NO_RA_Counter = 0
LineCounter = 1

DO WHILE NOT EOF(1)
  LINE INPUT #1, OpsDataLine
  IF TRIM$(PARSE$(OpsDataLine, ";", %FIELD_RetailAreaActual)) = "RX" OR _
  TRIM$(PARSE$(OpsDataLine, ";", %FIELD_RetailAreaActual)) = " " OR _
  TRIM$(PARSE$(OpsDataLine, ";", %FIELD_RetailAreaActual)) = "RY" THEN

    INCR NO_RA_Counter

  ELSE

    FOR i = 1 TO NumberOfRetailAreas
      IF TRIM$(PARSE$(OpsDataLine, ";", %FIELD_RetailAreaActual)) = TRIM$(aRetailArea(i,1)) THEN
        TempFactor = PARSE$(OpsDataLine, ";", %FIELD_FlightPAXDFfactor)
        REPLACE ", " WITH "." IN TempFactor 'just in case a comma instead of decimal point is used (
"121,6" --> "121.6")

        aRetailArea(i,2) = TRIM$(STR$( VAL(aRetailArea(i,2)) + VAL(TempFactor) )) 'sum up factors
        aRetailArea(i,3) = TRIM$(STR$(VAL(aRetailArea(i,3))+1)) 'increase counter
        aRetailArea(i,4) = TRIM$(STR$( ROUND(VAL(aRetailArea(i,2)) / VAL(aRetailArea(i,3)),5) )) 'calc
avg values instantly
        i = NumberOfRetailAreas
      END IF
    NEXT i

  END IF

  INCR LineCounter
  ProgressBoxUpdate INT(LineCounter/TotalLines*100)
WEND 'EOF(1)

CLOSE #1

```

```

ProgressBoxHide

ConsoleMessageBox "Number of retail areas not defined (RX or RY) = " +
TRIM$(STR$(NO_RA_Counter)),%DEFAULT,"INFO",%DEFAULT,0

'for transformation into factors

'determine smallest value as a baseline
MinValue = 999999.9
FOR i = 1 TO NumberOfRetailAreas
  IF VAL(aRetailArea(i,4)) < MinValue THEN
    MinValue = VAL(aRetailArea(i,4))
  END IF
NEXT i

'higher values will be a factor to the baseline value then

FOR i = 1 TO NumberOfRetailAreas
  aRetailArea(i,4) = TRIM$(STR$(ROUND(VAL(aRetailArea(i,4)) / MinValue, 5)))
NEXT i

'write suggested values into the retail area factor file as appended comment lines
OPEN FILE_RetailAreaFactors FOR APPEND AS #2
PRINT #2, " "
PRINT #2, "/"
PRINT #2, "// Suggested Retail Area Factors based on:"
PRINT #2, "// File: " + FileName
PRINT #2, "// Entry: " + DATE$ + " / " + TIME$
PRINT #2, "/"

FOR i = 1 TO UBOUND(aRetailArea(1))

  TempFactor = TRIM$(USING$("###.####", VAL(aRetailArea(i,4))))

  'REPLACE "." WITH "," IN TempFactor 'just in case a comma instead of decimal point is used ( "1.6" --> "1,6")

  PRINT #2, "/" + aRetailArea(i,1) + ";" + TempFactor + " [ Flights: " + aRetailArea(i,3) + " ]"

NEXT i
PRINT #2, "/"

CLOSE #2

lResult = ConsoleMessageBox("Suggested factors written. Do you want to edit file?", %YESNO+%HANDBOX+%DEFBUTTON1,
"INFO", %IDI_QUESTION, %FALSE)

IF lResult = %YESBUTTON THEN

  CALL LogEntry(FUNCNAME$, "START: EDIT/WATCH FACTORS")
  SHELL "notepad.exe " + FILE_RetailAreaFactors
  ConsoleMessageBox "If you have changed any Retail Area Factors, \n" + _
    "remember to update them in data file and\n" + _
    "to re-calc flight revenue.",%DEFAULT,"INFO",%DEFAULT,0
  CALL LogEntry(FUNCNAME$, "END: EDIT/WATCH FACTORS")

ELSE
  ConsoleMessageBox "For later edit use same function but only for edit then.",%DEFAULT,"INFO",%DEFAULT,0
END IF

CALL LogEntry(FUNCNAME$, "END")

cls

END SUB 'SuggestRetailAreaFactor()

'-----

SUB OpenBackGroundWindow()

LOCAL hBmp AS LONG
LOCAL h, w, hGW AS LONG
LOCAL lResult AS LONG

CALL LogEntry(FUNCNAME$, "START")

'ConsoleWindow %MINIMIZE

'w = 960
'h = 720
w = 806
h = 317

'GRAPHIC WINDOW "aeroCUBE: RESULTS", 30, 30, w, h TO hGW
'GRAPHIC WINDOW "aeroCUBE: RESULTS", 30, 30, w, h TO hGW

'GRAPHIC ATTACH hGW, 0&, REDRAW
'GRAPHIC COLOR RGB(0,0,0), RGB(255,255,255)
'GRAPHIC CLEAR

'GRAPHIC BITMAP LOAD PATH_APPLICATION+"aeroCUBE_back01.bmp", w, h TO hBmp

```

```

CURSOR OFF
BrushColor %BLACK
ConsoleGfx 0,0,0,0

    BrushColor %BLACK
GfxCls

GfxFontName "Arial"
GfxFontSize 20
DrawFrom 0,0
DrawTextRow "TEST TEST TEST", %TEXT_CENTER OR %TEXT_SHADOW

GfxWindow %GFX_SHOW

    'DisplayJpeg PATH_APPLICATION+"TerminalLayout.jpg"

    'DisplayImage PATH_APPLICATION+"TerminalLayout.jpg"
    lResult = StretchImage(PATH_APPLICATION+"TerminalLayout.jpg", 806, 317)

    'GRAPHIC COPY hBmp, 0 TO (1, 1)
    'GRAPHIC REDRAW

PRINT
PRINT "PRESS 'Q' TO QUIT GRAPHICS WINDOW."

DO
    IF UCASE$(WAITKEY$) = "Q" THEN
        EXIT DO
    END IF

WEND
cls

    'GRAPHIC BITMAP END
    'GRAPHIC WINDOW END

'ConsoleWindow %RESTORE

CALL LogEntry(FUNCNAME$, "END")

END SUB 'OpenBackgroundWindow()

'-----

SUB ConvertCommaToDecimalPoint(BYVAL FileName AS STRING)

'convert fields with comma character for decimal point to decimal point character
' 123,4 --> 123.4

CALL LogEntry(FUNCNAME$, "START")
'-----
LOCAL OpsDataLine AS STRING
LOCAL TempField AS STRING
LOCAL NumberOfRecords AS LONG
LOCAL LineCounter AS LONG
LOCAL ElementCounter AS INTEGER

'-----
OPEN FileName FOR INPUT AS #1
FILESCAN #1, RECORDS TO NumberOfRecords 'used for progress calculation
OPEN LEFT$(FileName, LEN(FileName)-4) + ".new" FOR OUTPUT AS #2

ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing "+TRIM$(USING$(###,###,###",
NumberOfRecords))+ " Records.", "Converting decimal: comma->point...", %FALSE

LineCounter = 1
DO WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    TempField = PARSE$(OpsDataLine, ";", %FIELD_FlightPAXDFfactor)
    REPLACE ANY "," WITH "." IN TempField
    OpsDataLine = StringUpdate(OpsDataLine, %FIELD_FlightPAXDFfactor, TempField)

    TempField = PARSE$(OpsDataLine, ";", %FIELD_RetailAreaFactorActual)
    REPLACE ANY "," WITH "." IN TempField
    OpsDataLine = StringUpdate(OpsDataLine, %FIELD_RetailAreaFactorActual, TempField)

    TempField = PARSE$(OpsDataLine, ";", %FIELD_RetailAreaFactorSeason)
    REPLACE ANY "," WITH "." IN TempField
    OpsDataLine = StringUpdate(OpsDataLine, %FIELD_RetailAreaFactorSeason, TempField)

    TempField = PARSE$(OpsDataLine, ";", %FIELD_RetailAreaFactorOpti)
    REPLACE ANY "," WITH "." IN TempField

```

```

OpsDataLine = StringUpdate(OpsDataLine, %FIELD_RetailAreaFactorOpti, TempField)

TempField = PARSE$(OpsDataLine, ";", %FIELD_RetailRevenueActual)
REPLACE ANY ", " WITH "." IN TempField
OpsDataLine = StringUpdate(OpsDataLine, %FIELD_RetailRevenueActual, TempField)

TempField = PARSE$(OpsDataLine, ";", %FIELD_RetailRevenueSeason)
REPLACE ANY ", " WITH "." IN TempField
OpsDataLine = StringUpdate(OpsDataLine, %FIELD_RetailRevenueSeason, TempField)

TempField = PARSE$(OpsDataLine, ";", %FIELD_RetailRevenueOpti)
REPLACE ANY ", " WITH "." IN TempField
OpsDataLine = StringUpdate(OpsDataLine, %FIELD_RetailRevenueOpti, TempField)

PRINT #2, OpsDataLine
ProgressBoxUpdate INT(LineCounter/NumberOfRecords*100)
INCR LineCounter

WEND 'EOF(1)

ProgressBoxHide

CLOSE #1
CLOSE #2

CALL ShiftFileIntoHistory(LEFT$(FileName, LEN(FileName)-4))
CALL LogEntry(FUNCNAME$, "END")

END SUB 'ConvertCommaToDecimalPoint()

'-----

SUB ReportGatesChanged()

'creates a file that contains for each day of the selected period
'the number of different entries (%FIELD_Gate_Actual vs. %FIELD_Gate_Season)

'--> date;
'total number of flights on that day;
'as before but with gate info available for both actual and seasonal;
'number of flights with gate change;
'gate change that resulted in diff. retail area
'--> 27.05.2006;1522;1244;190;23

CALL LogEntry(FUNCNAME$, "START")

'-----
LOCAL OpsDataLine AS STRING
LOCAL TempField AS STRING
LOCAL NumberOfRecords AS LONG
LOCAL LineCounter AS LONG
LOCAL DateTimeStamp AS STRING
LOCAL SelectedStartDate AS STRING
LOCAL SelectedEndDate AS STRING
LOCAL CurrentDate AS STRING
LOCAL ElementCounter AS INTEGER
LOCAL i AS INTEGER
LOCAL lDateMatch AS LONG

DIM aSeason(1 TO 2) AS STRING
LOCAL nSeason AS INTEGER

'-----

aSeason(1) = FILE_SUMMER
aSeason(2) = FILE_WINTER

SelectedStartDate = "2006-03-26"
SelectedStartDate = REMOVE$(ConsoleInputBox$(1, %CENTER, %CENTER, _
"Start date (YYYY-MM-DD)", _
"Gate changes", SelectedStartDate, 0, %FALSE), ANY "--")

IF LEN(SelectedStartDate) = 8 AND NOT ConsoleInputBoxCancel THEN
SelectedEndDate = "2007-03-24"
SelectedEndDate = REMOVE$(ConsoleInputBox$(1, %CENTER, %CENTER, _
"End date (YYYY-MM-DD)", _
"Gate changes", SelectedEndDate, 0, %FALSE), ANY "--")

IF LEN(SelectedEndDate) = 8 AND NOT ConsoleInputBoxCancel THEN

'generate an array with an entry for each day

ElementCounter = 1
CurrentDate = SelectedStartDate
DO WHILE CurrentDate <> SelectedEndDate AND ElementCounter < 1000
CurrentDate = NextDay(CurrentDate)
INCR ElementCounter
WEND 'array

```

```

IF ElementCounter < 1000 THEN

    DIM aDateGate (1 TO ElementCounter, 1 TO 5) AS STRING

    'initialize array

    CurrentDate = SelectedStartDate
    aDateGate(1, 1) = CurrentDate
    aDateGate(1, 2) = "0"
    aDateGate(1, 3) = "0"
    aDateGate(1, 4) = "0"
    aDateGate(1, 5) = "0"

    FOR i = 2 TO ElementCounter
        aDateGate(i, 1) = NextDay(CurrentDate)
        aDateGate(i, 2) = "0"
        aDateGate(i, 3) = "0"
        aDateGate(i, 4) = "0"
        aDateGate(i, 5) = "0"
        CurrentDate = NextDay(CurrentDate)
    NEXT i

    DateTimeStamp = TIME$
    REPLACE ANY ":" WITH "-" IN DateTimeStamp
    DateTimeStamp = DATE$ + "_" + DateTimeStamp
    OPEN PATH_DATA + $FILE_OUTPUT + "_GATECHANGE_" + DateTimeStamp + ".txt" FOR OUTPUT AS #2

    '--- as records are not sorted by e.g. ATD, try in each season ---

    FOR nSeason = 1 TO 2

        OPEN aSeason(nSeason) FOR INPUT AS #1
        FILESCAN #1, RECORDS TO NumberOfRecords 'used for progress calculation

        ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing
"+TRIM$(USING$( "###,###,###", NumberOfRecords))+ _
        " Records.", "Detecting gate changes...", %FALSE

        LineCounter = 1
        DO WHILE NOT EOF(1)

            LINE INPUT #1, OpsDataLine

            'ARRAY SCAN aDateGate(1,1) FOR ElementCounter+1, =LEFT$(PARSE$(OpsDataLine, ";",
%FIELD_ATD),8), TO lDateMatch

            lDateMatch = 0
            FOR i = 1 TO ElementCounter
                IF aDateGate(i,1) = LEFT$(PARSE$(OpsDataLine, ";", %FIELD_ATD),8) THEN
                    lDateMatch = i
                    i = ElementCounter
                END IF
            NEXT i

            IF lDateMatch <> 0 THEN 'the date has been located in array

                'increase count for: flights total
                aDateGate(lDateMatch, 2) = TRIM$(STR$(VAL(aDateGate(lDateMatch, 2)) + 1))

                IF PARSE$(OpsDataLine, ";", %FIELD_Gate_Actual) <> "" AND PARSE$(OpsDataLine, ";",
%FIELD_Gate_Season) <> "" THEN

                    'THEREOF: increase count for: flights which have both ACTUAL and SEASONAL gate info
                    aDateGate(lDateMatch, 3) = TRIM$(STR$(VAL(aDateGate(lDateMatch, 3)) + 1))

                    IF PARSE$(OpsDataLine, ";", %FIELD_Gate_Actual) <> PARSE$(OpsDataLine, ";",
%FIELD_Gate_Season) THEN

                        'THEREOF: increase count for: flights with different gates
                        aDateGate(lDateMatch, 4) = TRIM$(STR$(VAL(aDateGate(lDateMatch, 4)) + 1))

                        IF PARSE$(OpsDataLine, ";", %FIELD_RetailAreaActual) <> PARSE$(OpsDataLine,
";", %FIELD_RetailAreaSeason) _
                            AND PARSE$(OpsDataLine, ";", %FIELD_RetailAreaSeason) <> "RX" _
                            AND PARSE$(OpsDataLine, ";", %FIELD_RetailAreaSeason) <> "RY" THEN
                            'THEREOF: increase count for: flights with different retail areas
                            aDateGate(lDateMatch, 5) = TRIM$(STR$(VAL(aDateGate(lDateMatch, 5)) +
1))

                        'PRINT #2, OpsdataLine
                    END IF
                END IF
            END IF

            ProgressBoxUpdate INT(LineCounter/NumberOfRecords*100)
            INCR LineCounter

        WEND 'EOF(1)

        ProgressBoxHide

```



```

        CLOSE #1
    NEXT nSeason
    FOR i = 1 TO ElementCounter
        PRINT #2, aDateGate(i, 1) + ";" + aDateGate(i, 2) + ";" + aDateGate(i, 3) + ";" + aDateGate(i,
4) + ";" + aDateGate(i, 5)
    NEXT i
    CLOSE #2
    ConsoleMessageBox "Number of Days = " + STR$(ElementCounter)+ _
        "\n\nFor changed gates info see file:\n\n" + _
        $FILE_OUTPUT + "_GATECHANGE_" + DateTimeStamp + ".txt",%DEFAULT,"INFO",%DEFAULT,0
    ELSE 'ElementCounter is NOT <1000
        ConsoleMessageBox "Number of Days > " + STR$(ElementCounter-1)+ _
            "\n\nIt seems to be a mistake on input of dates." + _
            "\nNothing reported.",%DEFAULT,"INFO",%DEFAULT,0
    END IF
    ELSE
        ConsoleMessageBox "No end date chosen. Nothing reported. ",%DEFAULT,"INFO",%DEFAULT,0
    END IF
    ELSE
        ConsoleMessageBox "No start date chosen. Nothing reported. ",%DEFAULT,"INFO",%DEFAULT,0
    END IF 'SelectedStartDate
    CALL LogEntry(FUNCNAME$, "END")
END SUB 'ReportGatesChanged()

'-----
SUB ReportStatsPerDay()
'creates a file that contains for each day of the selected period
'the sum of the field chosen

'--> date;
'total number of flights on that day;
'sum of field
'avg of field
'--> 27.05.2006;1522;3456.33;0.53

CALL LogEntry(FUNCNAME$, "START")

'-----
LOCAL OpsDataLine AS STRING
LOCAL TempField AS STRING
LOCAL NumberOfRecords AS LONG
LOCAL LineCounter AS LONG
LOCAL DateTimeStamp AS STRING
LOCAL SelectedStartDate AS STRING
LOCAL SelectedEndDate AS STRING
LOCAL CurrentDate AS STRING
LOCAL ElementCounter AS INTEGER
LOCAL i AS INTEGER
LOCAL lDateMatch AS LONG
LOCAL ExportField AS INTEGER
LOCAL InfoField AS INTEGER

DIM aSeason(1 TO 2) AS STRING
LOCAL nSeason AS INTEGER

'-----
aSeason(1) = FILE_SUMMER
aSeason(2) = FILE_WINTER

InfoField = %FIELD_DestCountry

SelectedStartDate = "2006-03-26"
SelectedStartDate = REMOVE$(ConsoleInputBox$(1, %CENTER, %CENTER, _
    "ATD: Start date (YYYY-MM-DD)", _
    "Daily Stats", SelectedStartDate, 0, %FALSE), ANY "-")

IF LEN(SelectedStartDate) = 8 AND NOT ConsoleInputBoxCancel THEN
    SelectedEndDate = "2007-03-24"
    SelectedEndDate = REMOVE$(ConsoleInputBox$(1, %CENTER, %CENTER, _
        "ATD: End date (YYYY-MM-DD)", _
        "Daily Stats", SelectedEndDate, 0, %FALSE), ANY "-")

IF LEN(SelectedEndDate) = 8 AND NOT ConsoleInputBoxCancel THEN
    'generate an array with an entry for each day

```

```

ElementCounter = 1
CurrentDate = SelectedStartDate
DO WHILE CurrentDate <> SelectedEndDate AND ElementCounter < 1000
    CurrentDate = NextDay(CurrentDate)
    INCR ElementCounter
WEND 'array

IF ElementCounter < 1000 THEN

    DIM aDateGate (1 TO ElementCounter, 1 TO 5) AS STRING

    'initialize array

    CurrentDate = SelectedStartDate
    aDateGate(1, 1) = CurrentDate
    aDateGate(1, 2) = "0"
    aDateGate(1, 3) = "0"
    aDateGate(1, 4) = "0"
    aDateGate(1, 5) = "---" ' additional info (e.g. any field can be added here)

    FOR i = 2 TO ElementCounter
        aDateGate(i, 1) = NextDay(CurrentDate)
        aDateGate(i, 2) = "0"
        aDateGate(i, 3) = "0"
        aDateGate(i, 4) = "0"
        aDateGate(i, 5) = "---"
        CurrentDate = NextDay(CurrentDate)
    NEXT i

    ExportField = SelectField()

    IF ExportField >1 AND ExportField <37 THEN

        cls
        PRINT "selected Field = " + TRIM$(STR$(ExportField))

        DateTimeStamp = TIME$
        REPLACE ANY ":" WITH "-" IN DateTimeStamp
        DateTimeStamp = DATE$ + "_" + DateTimeStamp
        OPEN PATH_DATA + $FILE_OUTPUT + "_SUMAVG_DAY_Field_" + TRIM$(STR$(ExportField)) + "_" +
        DateTimeStamp + ".txt" FOR OUTPUT AS #2

        '--- as records are not sorted by e.g. ATD, try in each season ---

        FOR nSeason = 1 TO 2

            OPEN aSeason(nSeason) FOR INPUT AS #1
            FILESCAN #1, RECORDS TO NumberOfRecords 'used for progress calculation

            IF LEN(aSeason(nSeason)) >40 THEN
                PRINT "Basis: ATD in FILE: ..." + RIGHT$(aSeason(nSeason),40)
            ELSE
                PRINT "Basis: ATD in FILE: " + aSeason(nSeason)
            END IF

            ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing
            "+TRIM$(USING$(###,###,###, NumberOfRecords))+ _
            " Records.", "Detecting days ...", %FALSE

            LineCounter = 1
            DO WHILE NOT EOF(1)

                LINE INPUT #1, OpsDataLine

                lDateMatch = 0
                FOR i = 1 TO ElementCounter
                    IF aDateGate(i,1) = LEFT$(PARSE$(OpsDataLine, ";", %FIELD_ATD),8) THEN
                        lDateMatch = i
                        i = ElementCounter
                    END IF
                NEXT i

                IF lDateMatch <> 0 THEN 'the date has been located in array

                    'increase count for: flights total
                    aDateGate(lDateMatch, 2) = TRIM$(STR$(VAL(aDateGate(lDateMatch, 2)) + 1))

                    'sum up field for that day
                    'in case ExportField is 'DELAY' treat a '9999' as '0'
                    IF ExportField = %FIELD_DelayMinutes AND PARSE$(OpsDataLine, ";", ExportField) =
                    "9999" THEN
                        ' do not add anything
                    ELSE
                        aDateGate(lDateMatch, 3) = TRIM$(STR$(VAL(aDateGate(lDateMatch, 3)) + VAL(
                    PARSE$(OpsDataLine, ";", ExportField) )))
                    END IF

                    'calc avg for field for that day
                    aDateGate(lDateMatch, 4) = TRIM$(STR$(VAL(aDateGate(lDateMatch, 3)) /
                    VAL(aDateGate(lDateMatch, 2) )))

```

```

        'InfoField
        aDateGate(lDateMatch, 5) = PARSE$(OpsDataLine, ";", InfoField)

    END IF

    ProgressBoxUpdate INT((LineCounter/NumberOfRecords*100)
    INCR LineCounter

    WEND 'EOF(1)

    ProgressBoxHide

    CLOSE #1

    NEXT nSeason

    FOR i = 1 TO ElementCounter
        PRINT #2, aDateGate(i, 1) + ";" + aDateGate(i, 2) + ";" + aDateGate(i, 3) + ";" +
        aDateGate(i, 4) + ";" + aDateGate(i, 5)
    NEXT i

    CLOSE #2

    ConsoleMessageBox "Number of Days = " + STR$(ElementCounter)+ _
        "\n\nFor daily stats see file:\n\n" + _
        $FILE_OUTPUT + "_SUMAVG_DAY_Field_" + TRIM$(STR$(ExportField)) + "_" +
DateTimeStamp + ".txt",%DEFAULT,"INFO",%DEFAULT,0

    ELSE

        ConsoleMessageBox "No field chosen. Nothing reported.", %DEFAULT,"INFO",%DEFAULT,0

    END IF

    ELSE 'ElementCounter is NOT <1000

        ConsoleMessageBox "Number of Days > " + STR$(ElementCounter-1)+ _
            "\n\nIt seems to be a mistake on input of dates." + _
            "\nNothing reported.",%DEFAULT,"INFO",%DEFAULT,0

    END IF

    ELSE

        ConsoleMessageBox "No end date chosen. Nothing reported. ",%DEFAULT,"INFO",%DEFAULT,0

    END IF

    ELSE

        ConsoleMessageBox "No start date chosen. Nothing reported. ",%DEFAULT,"INFO",%DEFAULT,0

    END IF 'SelectedStartDate

    cls

    CALL LogEntry(FUNCNAME$, "END")

END SUB 'ReportStatsPerDay()

'-----

SUB GenerateJPGFiles()

    CALL LogEntry(FUNCNAME$, "START")

    '-----
    LOCAL OpsDataLine AS STRING
    LOCAL TimeInterval AS LONG
    LOCAL Gates AS INTEGER
    DIM GanttFiles(1 TO 7) AS STRING
    DIM FileWeekDay(1 TO 7) AS INTEGER
    LOCAL i AS INTEGER
    LOCAL j AS INTEGER
    LOCAL lResult AS LONG
    LOCAL SelectedDate AS STRING
    LOCAL StartDate AS STRING
    LOCAL SleepFactor AS INTEGER
    LOCAL FieldIndicator AS STRING
    LOCAL CatBIndicator AS STRING

    LOCAL TimeIndex AS STRING
    LOCAL LineCounter AS LONG
    LOCAL TotalLines AS LONG
    LOCAL lChangeDay AS INTEGER

    DIM FieldValue AS STRING
    DIM FieldCounter AS LONG

    DIM TimeIndex (1 TO 288) AS STRING
    LOCAL MyMouseOverX AS LONG
    LOCAL MyMouseOverY AS LONG
    LOCAL UserEvent AS STRING

    '-----

```

```

OnTimer 1, CODEPTR(MyGfxRefresh)

SelectedDate = "2006-03-26"

SelectedDate = REMOVE$(ConsoleInputBox$(1, %CENTER, %CENTER, _
    "Enter start date of week (Format: YYYY-MM-DD)", _
    "Select Week for Export", SelectedDate, 0, %FALSE), ANY "--")

IF LEN(SelectedDate) = 8 AND NOT ConsoleInputBoxCancel THEN
    'if a date has been entered

    '----- read time index for display purpose -----

        OPEN FILE_TIMEINDEX FOR INPUT AS #1

        LineCounter = 1
        WHILE NOT EOF(1)
            LINE INPUT #1, TimeIndex(LineCounter)
            INCR LineCounter
        WEND 'EOF(1)

        CLOSE #1

        'ConsoleMessageBox "TimeIndex has been read." & STR$(LineCounter),%DEFAULT,"INFO",%DEFAULT,0

    '----- read gate index for display purpose -----

        OPEN FILE_GATE_INFRA FOR INPUT AS #1

        FILESCAN #1, RECORDS TO TotalLines
        DIM GateArray(1 TO TotalLines+1, 1 TO 7) AS STRING

        LineCounter = 1
        WHILE NOT EOF(1)
            LINE INPUT #1, OpsDataLine
            IF LEFT$(OpsDataLine, 2) <> "//" THEN                                'bypass comment lines

                'parse the OpsDataLine and fill the field variables

                FieldValue = ""
                FieldCounter = 1

                FOR i = 1 TO LEN(OpsDataLine)
                    IF MID$(OpsDataLine, i,1) <> ";" THEN
                        'still in same field
                        FieldValue = FieldValue + MID$(OpsDataLine, i, 1)
                    ELSE
                        'field changes, so fill content into Gate Array
                        GateArray(LineCounter, FieldCounter) = FieldValue
                        INCR FieldCounter
                        FieldValue = ""
                    END IF
                NEXT i

                INCR LineCounter
            END IF
        WEND 'EOF(1)

        CLOSE #1

        'ConsoleMessageBox "GateIndex has been read." & STR$(LineCounter) + " " + STR$(UBOUND(GateArray(1))
        ),%DEFAULT,"INFO",%DEFAULT,0

    '-----

        GanttFiles(1) = FILE_GANTTVIEW + "_" + SelectedDate + ".TXT"
        StartDate = SelectedDate
        FileWeekDay(1) = DayWeek(SelectedDate)

        FOR i = 2 TO 7
            SelectedDate = NextDay(SelectedDate)
            GanttFiles(i) = FILE_GANTTVIEW + "_" + SelectedDate + ".TXT"
            FileWeekDay(i) = DayWeek(SelectedDate)
        NEXT i

        CURSOR OFF

        InitGraphicsTools 7, %GFX_TOOLSET_CONSOLE

        FOR i = 1 TO 7

            PRINT "Preparing Chart of Day " + TRIM$(STR$(i)) + " ..."

            UseGfxWindow i
            ConsoleGfx 17,0,80,24
            BrushColor %BLACK
            GfxCls

```

```

GfxFont "Arial", 23, 23, 5, %WHITE, 0, 90
DrawFrom 90, 280
DrawTextRow "GATES (A1..E26)", 0

GfxFont "Arial", 25, 25, 5, %WHITE, 0, 0
DrawFrom 520, 465
DrawTextRow "DAYTIME (0-24h)", 0

BrushColor %WHITE
DrawLine 80, 0, 80, 500

OPEN GanttFiles(i) FOR INPUT AS #1

TimeInterval = 1
WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    IF LEFT$(OpsDataLine, 2) <> "//" THEN

        FOR Gates = 1 TO PARSECOUNT(OpsDataLine, ANY ";")
            SELECT CASE PARSE$(OpsDataLine, ";", Gates)

                CASE "a"
                    'color green
                    BrushColor %GREEN

                CASE "b"
                    'color yellow
                    BrushColor %YELLOW

                CASE "c"
                    'color red
                    BrushColor %RED

            END SELECT

            IF PARSE$(OpsDataLine, ";", Gates) <> "" THEN
                DrawFrom TimeInterval*3+120, Gates*3+21
                DrawArea 3, 2
            END IF

        NEXT Gates
        INCR TimeInterval

    ELSE

        IF TRIM$(PARSE$(OpsDataLine, ":", 1)) = "// Field" THEN
            FieldIndicator = TRIM$(PARSE$(OpsDataLine, ":", 2))
        END IF

        IF TRIM$(PARSE$(OpsDataLine, ":", 1)) = "// Category B" THEN
            CatBIndicator = TRIM$(PARSE$(OpsDataLine, ":", 2))
        END IF

        END IF '// remark lines

    WEND 'EOF(1)
    CLOSE #1

NEXT i

cls

SleepFactor = 150

COLOR 0, 7
locate 1,1
PRINT "START ";
COLOR 7,0
PRINT " ";StartDate

COLOR 0, 7
locate 6,1
PRINT "FIELD ";
COLOR 7,0
PRINT LEFT$(FieldIndicator, 14)
PRINT MID$(FieldIndicator, 15, LEN(FieldIndicator)-14)

COLOR 0, 10
locate 9,1
PRINT "CAT A ";
COLOR 7,0
IF FieldIndicator = "Delay Minutes" THEN
    PRINT " < B"
ELSE
    PRINT " > B"
END IF

COLOR 0, 14
locate 10,1
PRINT "CAT B ";
COLOR 7,0
PRINT " " + CatBIndicator

COLOR 0, 12

```

```

locate 11,1
PRINT "CAT C ";
COLOR 7,0
IF FieldIndicator = "Delay Minutes" THEN
    PRINT " > B"
ELSE
    PRINT " < B"
END IF

COLOR 0, 7
locate 3,1
PRINT "R RUN "

locate 4,1
COLOR 7,0
PRINT "+/-";
COLOR 7,0
PRINT " ";TRIM$(STR$(SleepFactor));" "

COLOR 0, 7
locate 23,1
PRINT "Q QUIT "

lChangeDay = %FALSE
i = 1 'day to show

GfxWindow %GFX_HIDE
UseGfxWindow i
GfxWindow %GFX_SHOW

COLOR 0,7
locate 2,1
PRINT "D DAY ";
COLOR 7,0
PRINT " ";TRIM$(STR$(i))
locate 2,11
PRINT "WD ";TRIM$(STR$(FileWeekDay(i)));" "

DO

    UserEvent = inkey$

    IF lChangeDay = %TRUE THEN

        FOR j = 1 TO 5
            FOR i = 1 TO 7

                GfxWindow %GFX_HIDE
                UseGfxWindow i
                GfxWindow %GFX_SHOW

                COLOR 0,7
                locate 2,1
                PRINT "D DAY ";
                COLOR 7,0
                PRINT " ";TRIM$(STR$(i))
                locate 2,11
                PRINT "WD ";TRIM$(STR$(FileWeekDay(i)));" "

                SLEEP SleepFactor

            NEXT i
        NEXT j

        lChangeDay = %FALSE

    END IF

    IF LEN(UserEvent) = 4 AND ASC(UserEvent, 3) = 4 THEN

        'Select next day to view
        IF MouseY = 2 THEN
            IF MouseX >0 AND MouseX <8 THEN
                UserEvent = "D"
            END IF
        END IF

        'Toggle of auto-rotation between days
        IF MouseY = 3 THEN
            IF MouseX >0 AND MouseX <8 THEN
                UserEvent = "R"
            END IF
        END IF

        'QUIT
        IF MouseY = 23 THEN
            IF MouseX >0 AND MouseX <8 THEN
                UserEvent = "Q"
            END IF
        END IF

    END IF

locate 14,1

```

```

COLOR 0,7
PRINT "TIME ";
COLOR 7,0

CALL ValidMouseLocation (MyMouseOverX, MyMouseOverY)

IF INT( (MyMouseOverX-120)/3 ) > 0 AND INT( (MyMouseOverX-120)/3 ) <= 288 THEN
    PRINT " ";MID$(TimeIndex( INT((MyMouseOverX-120)/3) ),4,2 ) ;":"; RIGHT$(TimeIndex( INT((MyMouseOverX-
120)/3) ),2 ) ; " "
ELSE
    PRINT SPC(6)
END IF

locate 15,1
COLOR 0,7
PRINT "GATE ";
COLOR 7,0

CALL ValidMouseLocation (MyMouseOverX, MyMouseOverY)

IF INT((MyMouseOverY-21)/3) > 0 AND INT((MyMouseOverY-21)/3) < 153 THEN
    PRINT " "; GateArray( INT((MyMouseOverY-21)/3), 2 ); " "
ELSE
    PRINT SPC(6)
END IF

IF UCASE$(UserEvent) = "Q" THEN
    EXIT DO
END IF

IF UCASE$(UserEvent) = "D" THEN

    IF i >0 AND i < 7 THEN
        INCR i
    ELSE
        i = 1
    END IF

    GfxWindow %GFX_HIDE
    UseGfxWindow i
    GfxWindow %GFX_SHOW

    COLOR 0,7
    locate 2,1
    PRINT "D DAY ";
    COLOR 7,0
    PRINT " ";TRIM$(STR$(i))
    locate 2,11
    PRINT "WD ";TRIM$(STR$(FileWeekDay(i)));" "

END IF

IF UCASE$(UserEvent) = "R" THEN
    IF lChangeDay = %TRUE THEN
        lChangeDay = %FALSE
    ELSE
        lChangeDay = %TRUE
    END IF
END IF

IF UserEvent = "+" AND SleepFactor < 3000 THEN
    SleepFactor = SleepFactor + 50
    locate 4,1
    COLOR 7,0
    PRINT "+/-";
    COLOR 7,0
    PRINT " ";TRIM$(STR$(SleepFactor));" "
END IF

IF UserEvent = "-" AND SleepFactor >= 50 THEN
    SleepFactor = SleepFactor - 50
    locate 4,1
    COLOR 7,0
    PRINT "+/-";
    COLOR 7,0
    PRINT " ";TRIM$(STR$(SleepFactor));" "
END IF

WEND

FOR i = 1 TO 7

    UseGfxWindow i
    BrushColor %BLACK
    GfxCls

NEXT i

cls
ELSE
    ConsoleMessageBox "No valid date entered.",%DEFAULT,"INFO",%DEFAULT,0
END IF

```

```

    OnTimer 0, CODEPTR(MyGfxRefresh)

    CALL LogEntry(FUNCNAME$, "END")
END SUB 'GenerateJPGFiles()

'-----

SUB MyGfxRefresh()

    GfxRefresh 0
END SUB 'MyGfxRefresh()

'-----

SUB ValidMouseLocation (BYREF MyMouseOverX AS LONG, BYREF MyMouseOverY AS LONG)

    IF MouseOverX <> %GFX_NONE AND MouseOverX <> %GFX_BORDER_LEFT AND MouseOverX <> %GFX_BORDER_RIGHT AND _
        MouseOverX <> %GFX_BORDER_TOP AND MouseOverX <> %GFX_BORDER_BOTTOM THEN
        MyMouseOverX = MouseOverX
    ELSE
        MyMouseOverX = 1
    END IF

    IF MouseOverY <> %GFX_NONE AND MouseOverY <> %GFX_BORDER_LEFT AND MouseOverY <> %GFX_BORDER_RIGHT AND _
        MouseOverY <> %GFX_BORDER_TOP AND MouseOverY <> %GFX_BORDER_BOTTOM THEN
        MyMouseOverY = MouseOverY
    ELSE
        MyMouseOverY = 1
    END IF

END SUB 'ValidMouseLocation ()

'-----

FUNCTION DayWeek (BYVAL InDate AS STRING) AS INTEGER

    ' returns a number for the day of week
    ' implemented using Zeller's congruence formula
    ' Monday = 1 .. Sunday = 7

    LOCAL q AS SINGLE    'day of the month
    LOCAL m AS SINGLE    'month
    LOCAL K AS SINGLE    'year of the century
    LOCAL J AS SINGLE    'century
    LOCAL h AS SINGLE    'day of the week

    q = VAL(MID$(InDate, 7, 2))
    m = VAL(MID$(InDate, 5, 2))
    K = VAL(MID$(InDate, 3, 2))
    J = VAL(MID$(InDate, 1, 2))

    h = (q + INT( ((m+1)*26) / 10 ) + K + INT(K/4) + INT(J/4) + 5*J ) MOD 7

    'for ISO day week representation
    DayWeek = ( INT(h)+5) MOD 7 ) + 1

END FUNCTION 'DayWeek()

'-----

FUNCTION GetAlliance(BYVAL Airline2ltrCode AS STRING) AS STRING

    LOCAL i AS INTEGER

    GetAlliance = "---"

    FOR i = 1 TO UBOUND(gAirlineAlliances())
        IF PARSE$(gAirlineAlliances(i), ";", 2) = Airline2ltrCode THEN
            GetAlliance = PARSE$(gAirlineAlliances(i), ";", 1)
            EXIT FUNCTION
        END IF
    NEXT i

END FUNCTION 'IsInAlliance

'-----

SUB InitAlliances()

    LOCAL i AS INTEGER
    LOCAL LineCounter AS INTEGER
    LOCAL AllianceLine AS STRING

```



```

    FOR i = 1 TO UBOUND(gAirlineAlliances())
        gAirlineAlliances(i) = "---;--;---"
    NEXT i

    OPEN FILE AIRLINEALLIANCES FOR INPUT AS #1
    LineCounter = 0
    WHILE NOT EOF(1)
        LINE INPUT #1, AllianceLine
        IF LEFT$(AllianceLine, 2) <> "/" AND LEN(TRIM$(AllianceLine)) > 0 THEN
            INCR LineCounter
            gAirlineAlliances(LineCounter) = AllianceLine
        END IF
    WEND 'EOF(1)
    CLOSE #1
END SUB 'InitAlliances()

'-----

FUNCTION GetTerminalFromGate(BYREF Gate AS STRING) AS STRING

    SELECT CASE LEFT$(Gate, 1)
        CASE "A", "B", "C"
            GetTerminalFromGate = "1"
        CASE "D", "E"
            GetTerminalFromGate = "2"
        CASE ""
            GetTerminalFromGate = ""
        CASE ELSE
            ConsoleMessageBox "Invalid Gate-Info! Cannot determine corresponding Terminal!" + _
                "\nGATE: " + Gate, %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
            GetTerminalFromGate = "X"
        END SELECT
END FUNCTION 'GetTerminalFromGate()

'-----

FUNCTION GetCKIHallFromGate(BYREF Gate AS STRING) AS STRING

    SELECT CASE LEFT$(Gate, 1)
        CASE "A", "B", "C", "D", "E"
            GetCKIHallFromGate = LEFT$(Gate, 1)
        CASE ""
            GetCKIHallFromGate = ""
            ConsoleMessageBox "Invalid Gate-Info! Cannot determine corresponding CKI hall!" + _
                "\nGATE is: EMPTY-String! ", %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
        CASE ELSE
            ConsoleMessageBox "Invalid Gate-Info! Cannot determine corresponding CKI hall!" + _
                "\nGATE: " + Gate, %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
            GetCKIHallFromGate = "X"
        END SELECT
END FUNCTION 'GetCKIHallFromGate()

'-----

FUNCTION GetWingSpanCode(BYVAL AircraftType AS STRING) AS STRING

    LOCAL i AS INTEGER
    GetWingSpanCode = "0"
    FOR i = 1 TO gNumberOfAircraftTypes
        IF TRIM$(PARSE$(gDKGA_WSC(i), ";", 1)) = TRIM$(AircraftType) THEN
            GetWingSpanCode = TRIM$(PARSE$(gDKGA_WSC(i), ";", 2))
            EXIT FUNCTION
        END IF
    NEXT i
    ConsoleMessageBox "No WingSpanCode determined, because of no aircraft type found: " + AircraftType, _
        %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
END FUNCTION 'GetWingSpanCode()

```

```

-----
FUNCTION IsRemoteStand(BYVAL Stand AS STRING) AS INTEGER

    SELECT CASE LEFT$(TRIM$(Stand),1)

        CASE "A", "B", "C", "D", "E"
            IsRemoteStand = %FALSE

        CASE ""
            ConsoleMessageBox "Empty Stand-Info! Cannot determine whether CONTACT/REMOTE!",
                %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
            IsRemoteStand = %TRUE

        CASE ELSE
            IsRemoteStand = %TRUE

    END SELECT

END FUNCTION 'IsRemoteStand()

-----

SUB Show_DEBUG_COUNTERS()

    ConsoleMessageBox "DEBUG_COUNTER_1 = " + TRIM$(STR$(DEBUG_COUNTER_1)) + "\n" + _
        "DEBUG_COUNTER_2 = " + TRIM$(STR$(DEBUG_COUNTER_2)) + "\n" + _
        "DEBUG_COUNTER_3 = " + TRIM$(STR$(DEBUG_COUNTER_3)) + "\n" + _
        "DEBUG_COUNTER_4 = " + TRIM$(STR$(DEBUG_COUNTER_4)) + "\n" + _
        "DEBUG_COUNTER_5 = " + TRIM$(STR$(DEBUG_COUNTER_5)), _
        %OKONLY+%EXCLAMATIONBOX,"DEBUG COUNTER",%IDI_EXCLAMATION,0

END SUB

-----

SUB Reset_DEBUG_COUNTERS()

    DEBUG_COUNTER_1 = 0
    DEBUG_COUNTER_2 = 0
    DEBUG_COUNTER_3 = 0
    DEBUG_COUNTER_4 = 0
    DEBUG_COUNTER_5 = 0

END SUB

-----

SUB Dump_Into_DEBUG_File(BYREF aToBeDumped() AS STRING)

    'assumes that FILE_DEBUG is open at #99

    LOCAL i AS LONG

    FOR i = 1 TO UBOUND(aToBeDumped())
        PRINT #99, aToBeDumped(i)
    NEXT i

END SUB 'Dump_Into_DEBUG_File()

-----

```

```

SUB OPTI_Run()

LOCAL SelectedStartDate AS STRING
LOCAL SelectedEndDate AS STRING
LOCAL CurrentDay AS STRING
LOCAL DateTimeStamp AS STRING

CALL Reset_DEBUG_COUNTERS()

gSimRunRevenue = 0.00
gSimRunOppCost_A = 0.00
gSimRunOppCost_B = 0.00

ConsoleMessageBox "In case you run an optimization for SCENARIO analysis,\n" + _
"make sure that PRIOR to this run the following has been set accordingly:\n\n" + _
"In reference data:\n DF-Retail Factor\n Retail Area Factor\n Revenue Per Pax\n\n" + _
"In Data Cleaning:\n Update Retail Area Factor ACTUAL ops\n Update Retail Area Factor SEASONAL
planning" + _
"\n Calc Flight Revenue ACTUAL ops\n Calc Flight Revenue SEASONAL planning\n\n" + _
"IN CASE YOU STILL NEED TO CHANGE VALUES, JUST ABORT THE FOLLOWING DATE SELECTION." , _
%DEFAULT,"INFO",%DEFAULT,0

SelectedStartDate = "2006-03-26"
SelectedStartDate = REMOVE$(ConsoleInputBox$(1, LocOfCol(22), LocOfRow(5), _
"Start date (YYYY-MM-DD)", _
"OPTI Run", SelectedStartDate, 0, %FALSE), ANY "-")

IF LEN(SelectedStartDate) = 8 AND NOT ConsoleInputBoxCancel THEN
SelectedEndDate = "2007-03-24"
SelectedEndDate = REMOVE$(ConsoleInputBox$(1, LocOfCol(22), LocOfRow(5), _
"End date (YYYY-MM-DD)", _
"OPTI Run", SelectedEndDate, 0, %FALSE), ANY "-")

IF LEN(SelectedEndDate) = 8 AND NOT ConsoleInputBoxCancel THEN

CurrentDay = SelectedStartDate

CALL LogEntry(FUNCNAME$, "START: Edit Opti-Parameters")
SHELL "notepad.exe " + FILE_OPTIPARAMETERS
CALL LogEntry(FUNCNAME$, "END: Edit Opti-Parameters")

'open revenue-file
DateTimeStamp = TIMES
REPLACE ANY ":" WITH "-" IN DateTimeStamp
DateTimeStamp = DATES$ + "-" + DateTimeStamp
OPEN PATH_DATA + $FILE_REVENUES + "_" + DateTimeStamp + ".txt" FOR OUTPUT AS #3

OPEN PATH_APPLICATION+$FILE_DEBUG FOR APPEND AS #99

DO

CLS
PRINT "CurrentDay = "; CurrentDay

IF OPTI_FindSolution(CurrentDay) = %TRUE THEN
'ConsoleMessageBox "Day successfully allocated: " + CurrentDay , %DEFAULT,"INFO",%DEFAULT,0

'write day's schedule/plan into file
CALL OPTI_DumpDailyPlanIntoFile(CurrentDay)

CALL OPTI_DumpDailyAllocIntoFile(CurrentDay)

'update retail revenue stats
PRINT #3, CurrentDay + ";" + TRIM$(STR$(ROUND(gMaxFinalRevenue,2)))

gSimRunRevenue = gSimRunRevenue + gMaxFinalRevenue
gSimRunOppCost_A = gSimRunOppCost_A + gOppCostPerDay_A
gSimRunOppCost_B = gSimRunOppCost_B + gOppCostPerDay_B

ELSE
ConsoleMessageBox "Day could NOT be allocated!", %DEFAULT,"INFO",%DEFAULT,0
END IF

IF CurrentDay <> SelectedEndDate THEN
CurrentDay = NextDay(CurrentDay)
ELSE
EXIT DO
END IF

'ConsoleMessageBox "CurrentDay // SelectedEndDate: " + CurrentDay + " // " + SelectedEndDate,
%DEFAULT,"INFO",%DEFAULT,0

LOOP

'close revenue-file
CLOSE #3

ELSE
ConsoleMessageBox "No end date chosen. Nothing done. ",%DEFAULT,"INFO",%DEFAULT,0

```

```

END IF

ELSE

    ConsoleMessageBox "No start date chosen. Nothing done. ",%DEFAULT,"INFO",%DEFAULT,0

END IF 'SelectedStartDate

CLOSE #99

ConsoleMessageBox "End of optimization run.", %DEFAULT,"INFO",%DEFAULT,0
ConsoleMessageBox "Number of 'not-first-trials' = " + STR$(DEBUG_COUNTER_1) + _
    "\nNumber of shifts into next time interval = " + STR$(gNoFitCounter),
%DEFAULT,"INFO",%DEFAULT,0

ConsoleMessageBox "TotalRevenue for this run = " + USING$( "###,###,###.##", gSimRunRevenue)+ _
    "\nOppCost A for this run = " + USING$( "###,###,###.##", gSimRunOppCost_A) + _
    "\nOppCost B for this run = " + USING$( "###,###,###.##", gSimRunOppCost_B) + _
    "\nAs Percentage A = " + USING$( "###.##", ROUND((gSimRunOppCost_A/gSimRunRevenue*100),2)) + _
    "\nAs Percentage B = " + USING$( "###.##", ROUND((gSimRunOppCost_B/gSimRunRevenue*100),2)), _
    %OKONLY+%EXCLAMATIONBOX,"SIM-RUN STATS",%IDI_EXCLAMATION,0

END SUB 'OPTI_Run()

'-----

FUNCTION OPTI_FindSolution(BYVAL DKGA_Date AS STRING) AS INTEGER

    LOCAL lResult AS INTEGER
    LOCAL i AS LONG
    LOCAL j AS INTEGER
    LOCAL TempCounter AS INTEGER

    LOCAL TimeIntervalsPerDay AS INTEGER : TimeIntervalsPerDay = 288
    LOCAL SpecificTimeInterval AS INTEGER ' k (of Algorithm)
    LOCAL MaxNumberOfFlightsInInterval AS INTEGER : MaxNumberOfFlightsInInterval = %MaxDeparturesPerDay

    LOCAL OPTI_GatesAvailable AS STRING

    LOCAL OPTI_FlightsNotAllocatedInLastTimeInterval AS STRING
    LOCAL OPTI_FlightsToBeAllocatedInTimeInterval AS STRING
    LOCAL OPTI_Total_FlightsToBeAllocatedInTimeInterval AS STRING

    LOCAL OPTI_NumberOf_FlightsNotAllocatedInLastTimeInterval AS LONG
    LOCAL OPTI_NumberOf_FlightsToBeAllocatedInTimeInterval AS LONG
    LOCAL OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval AS LONG

    DIM OPTI_GateSet_PerFlight(1 TO 10, 1 TO 5) AS STRING 'dummy DIM. A REDIM is done below
                                                                (1 = FlightIndex _
                                                                '2 = GateIndex _
                                                                '3 = Retail Area
                                                                '4 = RA-Revenue _
                                                                '5 = Gate-Revenue

    Index _

    DIM OPTI_CombiResult_RA(1 TO gSolutionStackSize, 1 TO 2) AS STRING '(%SolutionStackSize Combi-Solutions are
    tested for picking gates according to retail areas) (1 = Retail Area Combination 2 = Revenue)

    LOCAL TempSum AS DOUBLE

    LOCAL lInsertTrial AS INTEGER
    LOCAL nTrial AS INTEGER
    LOCAL SolutionFlightIndex AS INTEGER
    LOCAL AssignedGate AS INTEGER
    LOCAL TempAllocatedGates AS STRING
    LOCAL TempOptiSingleRevenue AS STRING

    LOCAL lFound_TempOptiSingleRevenue AS INTEGER

    LOCAL InsertAttempt AS INTEGER

'-----

OPTI_FindSolution = %FALSE
g1SecondTryWithoutAllianceCompliance = %FALSE
gNoFitCounter = 0

'Initialize all relevant data: flight plan, gates infra, wing span codes, retail areas and related
CALL OPTI_Initialize(DKGA_Date)

'try to allocate all flights on that day

OPEN PATH_APPLICATION+$FILE_DEBUG FOR OUTPUT AS #99

LOCATE 3,1: PRINT "Generate day's gate allocation..."

PRINT "Start: "; TIME$

gMaxFinalRevenue = 0.00
gOppCostPerDay_A = 0.00
gOppCostPerDay_B = 0.00

```

```

'Dump_Into_DEBUG_File(gAllFlightsOnThatDay())
FOR SpecificTimeInterval = 1 TO TimeIntervalsPerDay

  RESET OPTI_CombiResult_RA()
  gSolutionCounter = 0

  LOCATE 6,1 : PRINT "SpecificTimeInterval: "; SpecificTimeInterval

  'set to null again
  OPTI_FlightsToBeAllocatedInTimeInterval = ""
  OPTI_NumberOf_FlightsToBeAllocatedInTimeInterval = 0
  OPTI_Total_FlightsToBeAllocatedInTimeInterval = ""
  OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval = 0

  'determine flights that need allocation in that time interval
  OPTI_FlightsToBeAllocatedInTimeInterval =
OPTI_Determine_FlightsToBeAllocatedInTimeInterval(SpecificTimeInterval)

  'determine number of flights that need allocation in that time interval
  IF LEN(TRIM$(OPTI_FlightsToBeAllocatedInTimeInterval)) > 0 THEN
    OPTI_NumberOf_FlightsToBeAllocatedInTimeInterval = PARSECOUNT(OPTI_FlightsToBeAllocatedInTimeInterval,
";")
  ELSE
    OPTI_NumberOf_FlightsToBeAllocatedInTimeInterval = 0
  END IF

  'second try without alliance rule
  IF g1SecondTryWithoutAllianceCompliance = %TRUE THEN
    OPTI_FlightsNotAllocatedInLastTimeInterval = ""
    OPTI_NumberOf_FlightsNotAllocatedInLastTimeInterval = 0
  END IF

  'cope also for the flights that could not be allocated in previous time interval (k-1)
  IF OPTI_NumberOf_FlightsNotAllocatedInLastTimeInterval > 0 THEN

    OPTI_Total_FlightsToBeAllocatedInTimeInterval = OPTI_FlightsNotAllocatedInLastTimeInterval + ";" +
OPTI_FlightsToBeAllocatedInTimeInterval

    IF LEFT$(OPTI_Total_FlightsToBeAllocatedInTimeInterval, 1) = ";" THEN
      OPTI_Total_FlightsToBeAllocatedInTimeInterval = RIGHT$(OPTI_Total_FlightsToBeAllocatedInTimeInterval,
LEN(OPTI_Total_FlightsToBeAllocatedInTimeInterval)-1)
    END IF

    'in case of empty OPTI_FlightsToBeAllocatedInTimeInterval there would be a semicolon at end --> remove it
    IF RIGHT$(OPTI_Total_FlightsToBeAllocatedInTimeInterval, 1) = ";" THEN
      OPTI_Total_FlightsToBeAllocatedInTimeInterval = LEFT$(OPTI_Total_FlightsToBeAllocatedInTimeInterval,
LEN(OPTI_Total_FlightsToBeAllocatedInTimeInterval)-1)
    END IF

  ELSE
    OPTI_Total_FlightsToBeAllocatedInTimeInterval = OPTI_FlightsToBeAllocatedInTimeInterval
  END IF

  'reset the values from previous interval (k-1)
  OPTI_FlightsNotAllocatedInLastTimeInterval = ""
  OPTI_NumberOf_FlightsNotAllocatedInLastTimeInterval = 0

  'determine no. of total flights to be allocated in this time interval

  IF LEN(TRIM$(OPTI_Total_FlightsToBeAllocatedInTimeInterval)) > 0 THEN
    OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval =
PARSECOUNT(OPTI_Total_FlightsToBeAllocatedInTimeInterval, ";")
  ELSE
    OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval = 0
  END IF

  'only continue if there are flights to be allocated in the SpecificTimeInterval
  IF OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval > 0 THEN

    'determine all available gates in this time interval (i.e. not occupied from previous assignments)
    OPTI_GatesAvailable = OPTI_Determine_AvailableGatesInInterval(SpecificTimeInterval)

    'for each flight in this interval: determine SET OF GATES that can be assigned to that flight
    '--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---
    REDIM OPTI_GateSet_PerFlight(1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval, 1 TO 5) AS
STRING
    '--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---

    FOR i = 1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval

      OPTI_GateSet_PerFlight(i, 1) = PARSE$(OPTI_Total_FlightsToBeAllocatedInTimeInterval, ";", i)
      'e.g. FlightIndex:      "303"

```

```

IF VAL(OPTI_GateSet_PerFlight(i, 1)) <> 0 THEN
    OPTI_GateSet_PerFlight(i, 2) = OPTI_Determine_EligibleGatesForFlight(
VAL(OPTI_GateSet_PerFlight(i, 1)), OPTI_GatesAvailable) 'e.g. GateIndex: "1;22;34;45;47"
ELSE
    ConsoleMessageBox "THIS SHOULD NOT OCCUR!!!\n\n" + _
        "FlightIndex = 0 !\n" + _
        "\nOPTI_GateSet_PerFlight(i, 1) :>>" +
OPTI_GateSet_PerFlight(i, 1) + "<<" + _
        "\nOPTI_Total_FlightsToBeAllocatedInTimeInterval :>>" +
OPTI_Total_FlightsToBeAllocatedInTimeInterval + "<<" + _
        "\nIndex i: " + STR$(i), _
        %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
END IF

'only if there are eligible gates ...
IF LEN(TRIM$(REMOVE$(OPTI_GateSet_PerFlight(i, 2), ";"))) > 0 THEN

    OPTI_GateSet_PerFlight(i, 3) =
OPTI_Determine_EligibleRetailAreasForFlight(OPTI_GateSet_PerFlight(i, 2)) 'e.g.
RetailAreaIndex: "1;3;4"
    OPTI_GateSet_PerFlight(i, 4) =
OPTI_Determine_RevenueForFlightInSpecificRetailArea(OPTI_GateSet_PerFlight(i, 1), OPTI_GateSet_PerFlight(i,3)) 'e.g.
Revenue: "234.12;22.90;1235.00"
    OPTI_GateSet_PerFlight(i, 5) =
OPTI_Determine_RevenueForFlightAtSpecificGates(OPTI_GateSet_PerFlight(i, 1), OPTI_GateSet_PerFlight(i, 2)) 'e.g.
Revenue: "234.12;234.12;234.12;22.90;22.90"

ELSE ' store for J2

    INCR gNoFitCounter

    OPTI_GateSet_PerFlight(i, 3) = "XXX"
    OPTI_GateSet_PerFlight(i, 4) = "XXX"
    OPTI_GateSet_PerFlight(i, 5) = "XXX"

    OPTI_FlightsNotAllocatedInLastTimeInterval = OPTI_FlightsNotAllocatedInLastTimeInterval + ";" +
OPTI_GateSet_PerFlight(i, 1)
    INCR OPTI_NumberOf_FlightsNotAllocatedInLastTimeInterval

    'remove semi-colon in case on first sign position
    IF LEFT$(OPTI_FlightsNotAllocatedInLastTimeInterval,1) = ";" THEN
        OPTI_FlightsNotAllocatedInLastTimeInterval =
RIGHT$(OPTI_FlightsNotAllocatedInLastTimeInterval, LEN(OPTI_FlightsNotAllocatedInLastTimeInterval)-1)
    END IF

END IF

NEXT i

'rewrite the array WITHOUT those flights for which no eligible gate could be determined

TempCounter = 0
FOR i = 1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval
    IF OPTI_GateSet_PerFlight(i, 3) <> "XXX" THEN

        INCR TempCounter

        OPTI_GateSet_PerFlight(TempCounter, 1) = OPTI_GateSet_PerFlight(i, 1)
        OPTI_GateSet_PerFlight(TempCounter, 2) = OPTI_GateSet_PerFlight(i, 2)
        OPTI_GateSet_PerFlight(TempCounter, 3) = OPTI_GateSet_PerFlight(i, 3)
        OPTI_GateSet_PerFlight(TempCounter, 4) = OPTI_GateSet_PerFlight(i, 4)
        OPTI_GateSet_PerFlight(TempCounter, 5) = OPTI_GateSet_PerFlight(i, 5)

    END IF

NEXT i

' clean the remainder of the old array content
FOR i = TempCounter + 1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval

    OPTI_GateSet_PerFlight(i, 1) = ""
    OPTI_GateSet_PerFlight(i, 2) = ""
    OPTI_GateSet_PerFlight(i, 3) = ""
    OPTI_GateSet_PerFlight(i, 4) = ""
    OPTI_GateSet_PerFlight(i, 5) = ""

NEXT i

'set new no. of total flights
OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval = TempCounter

'J1
'Opti run for flights that have not been allocated in previous time interval
'Opti run for flights with priority in current time interval

'--- COMBINATION STARTS HERE ... -----
LOCATE 7,1

```

```

PRINT "OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval: ";
OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval

LOCATE 8,1
PRINT "Flights in interval: ";
LOCATE 8,36
FOR i = 1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval
    PRINT OPTI_GateSet_PerFlight(i, 1) + " ";
NEXT i

LOCATE 9,1
PRINT "Number of EligibleGatesForFlight: ";
LOCATE 9,36
FOR i = 1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval
    PRINT USING$("###",PARSECOUNT(OPTI_GateSet_PerFlight(i, 2), ";")) + " ";
NEXT i

gBestRACombi = ""

IF OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval > 0 THEN

    gMaxTempRevenue = 0.00
    gMaxTheoRevenueRACombi = ""
    gMaxTheoreticalRevenue = OPTI_Determine_MaxTheoRevenue(OPTI_GateSet_PerFlight(),
OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval)

    'TRY THEO BEST COMBI FIRST -- IN CASE NOT POSSIBLE, DO COMBINATIONS
    FOR InsertAttempt = 1 TO 2

        '===== C O M B I (RECURSIVE CALL INITIATED HERE)=====

        IF InsertAttempt = 2 THEN

            '
            IF DEBUG_PRINT = %TRUE THEN
                ConsoleMessageBox "SpecificTimeInterval: " + STR$(SpecificTimeInterval) + _
                "\nStarting COMBI...", %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
            '
            END IF

            '--- start timer for defined termination of Combi here ---
            gTimer_CombiStart = INT(TIMER)
            gTimer_CombiStop = gTimer_CombiStart + %MaxCombiTime
            '-----

            '<===
            CALL OPTI_CombiTwoElements(OPTI_CombiResult_RA(), OPTI_GateSet_PerFlight(),
OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval, 1, "", "") '<===

            '<===

            '
            IF DEBUG_PRINT = %TRUE THEN
                ConsoleMessageBox "After combi done..." + _
                "\nnTrial = " + STR$(nTrial) + _
                "\nInsertAttempt = " + STR$(InsertAttempt) + _
                "\nSpecificTimeInterval: " + STR$(SpecificTimeInterval) + _
                "\ngBestRACombi: " + gBestRACombi + _
                "\ngSolutionStackSize: " + STR$(gSolutionStackSize), _
                %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
            '
            END IF

            END IF

            '--- TRY TO ASSIGN OPTIMUM GATE FOR ALL FLIGHTS ---

            FOR nTrial = 1 TO gSolutionStackSize

                IF InsertAttempt = 1 THEN
                    gBestRACombi = gMaxTheoRevenueRACombi 'has been determined in function:
OPTI_Determine_MaxTheoRevenue
                    gMaxTempRevenue = gMaxTheoreticalRevenue 'has been determined in function:
OPTI_Determine_MaxTheoRevenue
                    nTrial = 0
                ELSE

                    '
                    IF DEBUG_PRINT = %TRUE THEN
                        PRINT
                        FOR i = 1 TO gSolutionStackSize
                            PRINT i; " OPTI_CombiResult_RA(i, 1)= "; OPTI_CombiResult_RA(i, 1); " revenue = ";
OPTI_CombiResult_RA(i, 2)
                        NEXT i
                        ConsoleMessageBox "gBestRACombi wurde befüllt:" + _
                        "\nnTrial= " + STR$(nTrial) + _
                        "\nOPTI_CombiResult_RA(nTrial, 1)= " + OPTI_CombiResult_RA(nTrial,
1), _
                        %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
                    '
                    END IF

                    gBestRACombi = OPTI_CombiResult_RA(nTrial, 1)

                END IF

            END IF

            gBestRACombi = OPTI_CombiResult_RA(nTrial, 1)

            END IF

```

```

CALL OPTI_GateTimeMatrix_Transaction_Start()

IF TRIM$(REMOVE$(gBestRACombi, ";")) <> "" THEN 'only, if there's a value in gBestRACombi

    TempAllocatedGates = ""
    lInsertTrial = %TRUE

    FOR i = 1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval
        '.....CandidateRetailAreaIndex.....FlightIndex
        ..... TimeFrom .....
        AssignedGate = OPTI_GateChosen(VAL(PARSE$(gBestRACombi, ";", i)), i,
        SpecificTimeInterval, OPTI_GateSet_PerFlight() )

        IF AssignedGate = 0 THEN

            'so no gate has been found for that flight ('i')
            'this means that not the whole sequence could be allocated
            'an thus this nTrial has no valid solution

            lInsertTrial = %FALSE

            EXIT FOR 'so that a next best solution can be tested for free gates
        ELSE
            TempAllocatedGates = TempAllocatedGates + ";" + TRIM$(STR$(AssignedGate))
        END IF

    NEXT i 'try for next flight...

    'at this point all flights have been tested for insert into the matrix

    'remove semi-colon in case on first sign position
    IF LEFT$(TempAllocatedGates, 1) = ";" THEN
        TempAllocatedGates = RIGHT$(TempAllocatedGates, LEN(TempAllocatedGates)-1)
    END IF

    IF lInsertTrial = %TRUE THEN

        IF InsertAttempt > 1 THEN
            INCR DEBUG_COUNTER_1

            gOppCostPerDay_A = gOppCostPerDay_A + (gMaxTheoreticalRevenue-
            VAL(OPTI_CombiResult_RA(nTrial, 2)) )
            gOppCostPerDay_B = gOppCostPerDay_B + (VAL(OPTI_CombiResult_RA(1, 2))-
            VAL(OPTI_CombiResult_RA(nTrial, 2)) )

            END IF

            'sum up daily total revenue
            gMaxFinalRevenue = gMaxFinalRevenue + gMaxTempRevenue

            'and now flush/reset the transaction stack, because not necessary anymore
            CALL OPTI_GateTimeMatrix_Transaction_Commit()

            'for next time interval block those retail areas that have too many PAX
            CALL OPTI_Avoid_RetailArea_PAX_OverLoad(SpecificTimeInterval)

            InsertAttempt = 3 'i.e. exit also InsertAttempt-LOOP

            EXIT FOR '(nTrial-LOOP) this means no more next-best solution needs to be looked
at

        ELSE '---> lInsertTrial = %FALSE

            IF gGateTimeMatrix_Transaction_Stack_Counter > 0 THEN
                CALL OPTI_GateTimeMatrix_Transaction_Rollback()
            END IF

            END IF 'lInsertTrial = %TRUE

        ELSE 'gBestRACombi <> ""

            'in case a combination has been done and no
            'more values are in solution stack: exit rest of trials
            EXIT FOR

        END IF 'gBestRACombi <> ""

        IF InsertAttempt = 1 THEN
            EXIT FOR
        END IF

    NEXT nTrial

NEXT InsertAttempt

```



```

'if after all trials still no solution, an alternative methods
'needs to be applied or flights shifted into next interval

IF lInsertTrial = %FALSE THEN

    IF glSecondTryWithoutAllianceCompliance = %FALSE THEN

        PRINT #99, DKGA_Date + ";" + TRIM$(STR$(SpecificTimeInterval))

        glSecondTryWithoutAllianceCompliance = %TRUE
        SpecificTimeInterval = SpecificTimeInterval - 1
        'try it again with

different mode

    ELSE

        ConsoleMessageBox "Even no success without alliance rule!\nin TimeInterval: " +
STR$(SpecificTimeInterval), _
            %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
        glSecondTryWithoutAllianceCompliance = %FALSE

        'store for next interval's J1
        'BUT ONLY A SELECTED FLIGHT! NOT ALL!!!
        '...
        '...
        '...
        FOR i = 1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval
            OPTI_FlightsNotAllocatedInLastTimeInterval = OPTI_FlightsNotAllocatedInLastTimeInterval
+ ";" + OPTI_GateSet_PerFlight(i, 1)
            INCR OPTI_NumberOf_FlightsNotAllocatedInLastTimeInterval
        NEXT i

        'remove semi-colon in case on first sign position
        IF LEFT$(OPTI_FlightsNotAllocatedInLastTimeInterval,1) = ";" THEN
            OPTI_FlightsNotAllocatedInLastTimeInterval =
RIGHT$(OPTI_FlightsNotAllocatedInLastTimeInterval, LEN(OPTI_FlightsNotAllocatedInLastTimeInterval)-1)
        END IF

        OPTI_FindSolution = %TRUE 'means here that there was shift to next time interval

        ConsoleMessageBox "TotalRevenue for this day = " + USING$("###,###,###.##", gMaxFinalRevenue)+
-
            "\nOppCost A for this day = " + USING$("###,###,###.##", gOppCostPerDay_A) +
-
            "\nOppCost B for this day = " + USING$("###,###,###.##", gOppCostPerDay_B) +
-
            "\nAs Percentage A = " + USING$("###.##",
ROUND((gOppCostPerDay_A/gMaxFinalRevenue*100),2)) + _
            "\nAs Percentage B = " + USING$("###.##",
ROUND((gOppCostPerDay_B/gMaxFinalRevenue*100),2)), _
            %OKONLY+%EXCLAMATIONBOX,"DAILY STATS",%IDI_EXCLAMATION,0

        EXIT FUNCTION 'EXIT WHILE TESTING   @@@

    END IF

ELSE 'all flights have been allocated

    'update gAllFlightsOnThatDay() with:
    ' - %FIELD_Gate_Opti
    ' - %FIELD_RetailAreaOpti
    ' - %FIELD_RetailAreaFactorOpti
    ' - %FIELD_RetailRevenueOpti

    'glSecondTryWithoutAllianceCompliance = %FALSE

    FOR i = 1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval

        '%FIELD_Gate_Opti
        gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))) = StringUpdate(
gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))), _
            %FIELD_Gate_Opti, _

OPTI_Geno2Pheno_Gate(PARSE$(TempAllocatedGates, ";", i)) _
        )

        '%FIELD_RetailAreaOpti
        gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))) = StringUpdate(
gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))), _
            %FIELD_RetailAreaOpti,
-
            PARSE$(gaRetailArea(
GetRetailAreaIndexFromGateIndex(VAL(PARSE$(TempAllocatedGates, ";", i))), ";",1) _
        )

        '%FIELD_RetailAreaFactorOpti
        gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))) = StringUpdate(
gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))), _

```

```

%FIELD_RetailAreaFactorOpti, _
GetRetailFactor(PARSE$(gaRetailArea( GetRetailAreaIndexFromGateIndex(VAL(PARSE$(TempAllocatedGates, ";", i))) ),
";",1), gaRF() ) _
)
'
%FIELD_RetailRevenueOpti
TempOptiSingleRevenue = ""
lFound_TempOptiSingleRevenue = %FALSE
FOR j = 1 TO PARSECOUNT(OPTI_GateSet_PerFlight(i, 2), ";")
IF TRIM$(PARSE$(OPTI_GateSet_PerFlight(i, 2), ";", j)) = TRIM$(PARSE$(TempAllocatedGates,
";", i)) THEN
TempOptiSingleRevenue = PARSE$(OPTI_GateSet_PerFlight(i, 5), ";", j)
lFound_TempOptiSingleRevenue = %TRUE
EXIT FOR
END IF
NEXT j
IF TRIM$(TempOptiSingleRevenue) = "" THEN
IF lFound_TempOptiSingleRevenue = %TRUE THEN
ConsoleMessageBox "All flights allocated in SpecificTimeInterval: " +
STR$(SpecificTimeInterval) + _
"\nNumber Of Flights: " +
STR$(OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval) + _
"\nTempOptiSingleRevenue -->" + TempOptiSingleRevenue + "<---" + _
"\ngAllFlightsOnThatDay(): " +
gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))) , _
%OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
ELSE
ConsoleMessageBox "lFound_TempOptiSingleRevenue = %FALSE !!!" + _
"\n\nSpecificTimeInterval: " + STR$(SpecificTimeInterval) + _
"\nNumber Of Flights: " +
STR$(OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval) + _
"\nPARSE$(TempAllocatedGates, ';', i) " +
PARSE$(TempAllocatedGates, ";", i) + _
"\nTempOptiSingleRevenue -->" + TempOptiSingleRevenue + "<---" + _
"\ngAllFlightsOnThatDay(): " +
gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))) , _
%OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
END IF
END IF
gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))) = StringUpdate(
gAllFlightsOnThatDay(VAL(OPTI_GateSet_PerFlight(i, 1))), _
%FIELD_RetailRevenueOpti, _
TempOptiSingleRevenue
)
NEXT i
END IF
END IF 'OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval >= 1
'-----
'---- COMBINATION ENDS HERE ... -----
'-----
ELSE
PRINT #99, " NO ALLOCATIONS DUE IN THIS TIME INTERVAL"
END IF
NEXT SpecificTimeInterval
LOCATE 3, 35 : PRINT "finished."
LOCATE 4, 20 : PRINT "Stop: "; TIME$
'
ConsoleMessageBox "TotalRevenue for this day = " + USING$("###,###,###.##",
gMaxFinalRevenue)+ _
"\nOppCost A for this day = " + USING$("###,###,###.##", gOppCostPerDay_A)
+ _
"\nOppCost B for this day = " + USING$("###,###,###.##", gOppCostPerDay_B)
+ _
"\nAs Percentage A = " + USING$("###.##",
ROUND((gOppCostPerDay_A/gMaxFinalRevenue*100),2)) + _
"\nAs Percentage B = " + USING$("###.##",
ROUND((gOppCostPerDay_B/gMaxFinalRevenue*100),2)), _

```

```

'                                     %OKONLY+%EXCLAMATIONBOX,"DAILY STATS",%IDI_EXCLAMATION,0

OPTI_FindSolution = %TRUE

'CALL OPTI_GateTimeMatrix_DEBUG()

END FUNCTION 'OPTI_FindSolution()

-----

SUB OPTI_CombiTwoElements( BYREF OPTI_CombiResult_RA() AS STRING, _
                        BYREF OPTI_GateSet_PerFlight() AS STRING, _
                        BYVAL MaxNumberOfFlights AS INTEGER, _
                        BYVAL CurrentFlight AS INTEGER, _
                        BYVAL SolutionString AS STRING, _
                        BYVAL SolutionRAs AS STRING)

LOCAL i AS INTEGER
LOCAL j AS INTEGER
LOCAL TempString AS STRING
LOCAL TempSum AS DOUBLE
LOCAL TempRAs AS STRING

TempString = SolutionString
TempRAs = SolutionRAs

'-----
' ACTIVATE IF: Combi produces too many Solutions
'-----
' IF gSolutionCounter > %MaxSolutionsToBeTested THEN
'   CurrentFlight = MaxNumberOfFlights + 1
' END IF
'-----

' ACTIVATE IF: Combi takes too much time (terminate by timer)
'-----
IF INT(TIMER) > gTimer_CombiStop THEN
  CurrentFlight = MaxNumberOfFlights + 1
END IF
'-----

IF CurrentFlight <= MaxNumberOfFlights AND _
TRIM$(OPTI_GateSet_PerFlight(CurrentFlight, 1)) <> "" THEN

  IF TRIM$(REMOVE$(OPTI_GateSet_PerFlight(CurrentFlight, 3), ";")) <> "" THEN

    FOR i = 1 TO PARSECOUNT(OPTI_GateSet_PerFlight(CurrentFlight, 4), ";")
      TempString = TempString + PARSE$(OPTI_GateSet_PerFlight(CurrentFlight, 4), ";", i) + ";" 'to
store revenues
      TempRAs = TempRAs + PARSE$(OPTI_GateSet_PerFlight(CurrentFlight, 3), ";", i) + ";" 'to
store Retail Areas

      '+++ recursive call of procedure +++
      CALL OPTI_CombiTwoElements(OPTI_CombiResult_RA(), OPTI_GateSet_PerFlight(), MaxNumberOfFlights,
CurrentFlight+1, TempString, TempRAs)

      TempString = SolutionString
      TempRAs = SolutionRAs

    NEXT i

  END IF

  ELSE ' a next solution has been built

    'remove semicolons at end of solutions strings

    IF RIGHT$(TempString, 1) = ";" THEN
      TempString = LEFT$(TempString, LEN(TempString)-1) 'remove the last semicolon
    END IF

    IF RIGHT$(TempRAs, 1) = ";" THEN
      TempRAs = LEFT$(TempRAs, LEN(TempRAs)-1) 'remove the last semicolon
    END IF

    'now determine revenue of solution

    TempSum = 0
    FOR i = 1 TO PARSECOUNT(TempString, ";")
      TempSum = TempSum + VAL(PARSE$(TempString, ";", i))
    NEXT i

    'if revenue of solution is better than last best revenue
    'OR
    'Solution Stack not yet completely filled: store the result

    IF (TempSum >= gMaxTempRevenue) OR (gSolutionCounter < gSolutionStackSize) THEN

```

```

'insert only if number of free gates in retail area
'is not less than occurrences of that retail area

IF OPTI_IsValidRACombi(TempRAs, OPTI_GateSet_PerFlight()) = %TRUE THEN

    'store flight/RA combi to pick a gate in that RA later on
    gBestRACombi = TempRAs

    'store max revenue of that combi
    gMaxTempRevenue = TempSum

    'store best (gSolutionStackSize) solutions for later gate picking
    INCR gSolutionCounter
    'LOCATE 9,1 : PRINT SPC(30)
    'LOCATE 9,1 : PRINT "gSolutionCounter = " ; gSolutionCounter

    FOR i = 1 TO gSolutionStackSize
        IF TempSum >= VAL(OPTI_CombiResult_RA(i,2)) THEN

            'shift and insert into 2-DIM array
            FOR j = gSolutionStackSize TO i+1 STEP -1
                OPTI_CombiResult_RA(j,1) = OPTI_CombiResult_RA(j-1,1)
                OPTI_CombiResult_RA(j,2) = OPTI_CombiResult_RA(j-1,2)
            NEXT j

            OPTI_CombiResult_RA(i,1) = TempRAs
            OPTI_CombiResult_RA(i,2) = TRIM$(STR$(TempSum))

            EXIT FOR
        END IF
    NEXT i

ELSE

    TempSum = 0
    TempRAs = ""

    END IF 'OPTI_IsValidRACombi(TempRAs, OPTI_GateSet_PerFlight()) = %TRUE

END IF 'TempSum >= gMaxTempRevenue

END IF 'CurrentFlight <= MaxNumberOfFlights AND TRIM$(OPTI_GateSet_PerFlight(CurrentFlight, 1)) <> ""

END SUB 'OPTI_CombiTwoElements()

'-----

SUB OPTI_Initialize(BYVAL DKGA_Date AS STRING)

    'initializes (global) Opti variables

    LOCAL CurrentPos AS LONG
    LOCAL OpsDataLine AS STRING
    LOCAL LineCounter AS LONG

    LOCAL TempSGT AS INTEGER
    LOCAL Index_1 AS INTEGER
    LOCAL Index_2 AS INTEGER

    LOCAL GateIndex AS INTEGER
    LOCAL TimeIndex AS INTEGER

    DIM DKGA_SelectedFlightRecsOrg(1 TO %MaxDeparturesPerDay) AS STRING

    '-----

    'add field STAG= STD-SGT 'STAG = Scheduled Time at Gate

    '-----
    '--- PREPARE OPTI VARIABLES (FROM FILES) ---
    '-----

    '--- Initialize Optimization Parameters -----
    CALL OPTI_InitializeOptiParamFromFile()

    '--- Read Gate Infra for later use -----

    OPEN FILE_GATE_INFRA FOR INPUT AS #1
    LineCounter = 1

    WHILE NOT EOF(1)
        LINE INPUT #1, OpsDataLine
        IF LEFT$(OpsDataLine, 2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN
            gDKGA_GatesInfra(LineCounter) = OpsDataLine
            INCR LineCounter
        END IF
    END WHILE

```

```

        END IF
    WEND 'eof(1)
    CLOSE #1
    gNumberOfGates = LineCounter - 1

'--- Read WingSpanCodes for later use -----
OPEN FILE_WSC FOR INPUT AS #1
LineCounter = 1

WHILE NOT EOF(1)
    LINE INPUT #1, OpsDataLine
    IF LEFT$(OpsDataLine, 2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN
        gDKGA_WSC(LineCounter) = OpsDataLine
        INCR LineCounter
    END IF
WEND 'eof(1)
CLOSE #1

'--- Read RetailAreaDef for later use -----

'first count number of defined retail areas and then define array and read them

OPEN FILE_RETAIL_AREA_DEF FOR INPUT AS #1

LineCounter = 1
WHILE NOT EOF(1)
    LINE INPUT #1, OpsDataLine
    IF LEFT$(OpsDataLine,2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN 'no comment line in data file
        INCR LineCounter
    END IF
WEND 'eof(1)
CLOSE #1

'--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---
REDIM gaRetailArea(1 TO LineCounter) AS STRING
REDIM gRetailAreaNumberOfFreeGates(1 TO LineCounter) AS INTEGER
'--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---

'now read areas
OPEN FILE_RETAIL_AREA_DEF FOR INPUT AS #1
LineCounter = 1
WHILE NOT EOF(1)
    LINE INPUT #1, OpsDataLine
    IF LEFT$(OpsDataLine,2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN 'no comment line in data file
        gaRetailArea(LineCounter) = OpsDataLine
        INCR LineCounter
    END IF
WEND 'eof(1)
CLOSE #1

'--- Read all Retail Factor data into array for later use -----

OPEN FILE_RetailAreaFactors FOR INPUT AS #1
FILESIZE #1, RECORDS TO LineCounter
'--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---
REDIM gaRF(LineCounter,2) AS STRING
'--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---

LineCounter = 1
WHILE NOT EOF(1)
    LINE INPUT #1, OpsDataLine
    IF LEFT$(OpsDataLine,2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN 'no comment line in data file
        gaRF(LineCounter,1) = LEFT$(OpsDataLine, 2)
        gaRF(LineCounter,2) = RIGHT$(OpsDataLine, (LEN(OpsDataLine) - INSTR(OpsDataLine, ANY ";"))) 'get RF
factor
        INCR LineCounter
    END IF
WEND 'EOF(1)
CLOSE #1

'--- DF-Retail-Factor data for later use -----

OPEN FILE_DF_RETAIL_FACTOR FOR INPUT AS #1          'File with DF->RetailFactor value

WHILE NOT EOF(1)
    LINE INPUT #1, OpsDataLine
    'the first non-comment-line in file is to be the RevPerPax value
    IF LEFT$(OpsDataLine,2) <> "/" AND LEN(TRIM$(OpsDataLine)) > 0 THEN 'no comment line in data file
        REPLACE ",", " WITH "." IN OpsDataLine 'just in case a comma instead of decimal point is used ( "3,40" -->
"3.40")
        gDF_RetailFactor = VAL(OpsDataLine)
    END IF
WEND 'EOF(1)
CLOSE #1

'START: read avg. PAX-DF-Revenue for later use -----
--

OPEN FILE_RevPerPax FOR INPUT AS #1          'File with Average (DF) Revenue Per PAX

WHILE NOT EOF(1)

```

```

LINE INPUT #1, OpsDataLine

'the first non-comment-line in file is to be the RevPerPax value
IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file

    REPLACE "," WITH "." IN OpsDataLine 'just in case a comma instead of decimal point is used ( "5,80" -->
"5.80")

    gRevPerPax = VAL(OpsDataLine)

END IF

WEND 'EOF(1)

CLOSE #1

'END: read avg. PAX-DF-Revenue for later use -----

'--- Read the flight plan data -----
'this procedure is supposed to read a flight plan -- in this case here a past day's actual traffic is taken

PRINT "Reading day's records from file(s)...";

LineCounter = 1
CLOSE #1
OPEN FILE_SUMMER FOR INPUT AS #1

WHILE NOT EOF(1)

    LINE INPUT #1, OpsDataLine

    'store only in array if STD-Date matches the selected date
    '!!! DIFFERENCE TO SUB REPORTSSTATSPERDAY (which is based on ATD not STD) !!!

    IF LEFT$(PARSE$(OpsDataLine, ";", %FIELD_STD),8) = DKGA_Date THEN

        'Attach new field to flight record (position 37): TimeIndex
        OpsDataLine = OpsDataLine + TRIM$(STR$(GetTimeIndexFromTime(MID$(PARSE$(OpsDataLine, ";", %FIELD_STD),9,4)
)))

        'Attach new field to flight record (position 38): STAG (Scheduled Time At Gate: STD - SGT)

        'if there's no Standard Ground Time OR a very short one, take the default one that has been read during
initialization
        IF VAL(PARSE$(OpsDataLine, ";", %FIELD_StdGroundTime)) < gDKGA_MinutesAtGateMINIMUM THEN
            TempSGT = gDKGA_MinutesAtGateMINIMUM
        ELSE
            TempSGT = VAL(PARSE$(OpsDataLine, ";", %FIELD_StdGroundTime))
        END IF

        'now calculate the the STAG (= STD - SGT.Minutes) i.e.: %FIELD_STAG
        Index_1 = VAL(PARSE$(OpsDataLine, ";", %FIELD_TimeIndex))
        Index_2 = GetTimeIndexFromTime(GetTimeFromMinutes(TempSGT))

        IF Index_1 - Index_2 <=0 THEN
            OpsDataLine = OpsDataLine + ";0000"
        ELSE
            OpsDataLine = OpsDataLine + ";" + GetTimeFromTimeIndex(Index_1 - Index_2 + 1)
        END IF

        'store the result
        DKGA_SelectedFlightRecsOrg(LineCounter)= OpsDataLine

        INCR LineCounter

    END IF

WEND 'EOF(1)

CLOSE #1

'if no record found in summer season, try winter season... -----

IF LineCounter = 1 THEN

    PRINT "finished."
    PRINT "=>No record found in summer season. -- Trying winter season...";

    OPEN FILE_WINTER FOR INPUT AS #1

    WHILE NOT EOF(1)

        LINE INPUT #1, OpsDataLine

        'store only in array if STD-Date matches the selected date
        '!!! DIFFERENCE TO SUB REPORTSSTATSPERDAY (which is based on ATD not STD) !!!

```

```

IF LEFT$(PARSE$(OpsDataLine, ";", %FIELD_STD),8) = DKGA_Date THEN

  'Attach new field to flight record (position 37): TimeIndex
  OpsDataLine = OpsDataLine + TRIM$(STR$(GetTimeIndexFromTime(MID$(PARSE$(OpsDataLine, ";",
%FIELD_STD),9,4) )))

  'Attach new field to flight record (position 38): STAG (Scheduled Time At Gate: STD - SGT)

  'if there's no Standard Ground Time, take the default one that has been read during Initialization
  IF VAL(PARSE$(OpsDataLine, ";", %FIELD_StdGroundTime)) = 0 THEN
    TempSGT = gDKGA_MinutesAtGateMINIMUM
  ELSE
    TempSGT = VAL(PARSE$(OpsDataLine, ";", %FIELD_StdGroundTime))
  END IF

  'now calculate the the STAG (= STD - SGT.Minutes)
  Index_1 = VAL(PARSE$(OpsDataLine, ";", %FIELD_TimeIndex))
  Index_2 = GetTimeIndexFromTime(GetTimeFromMinutes(TempSGT))

  IF Index_1 - Index_2 <=0 THEN
    OpsDataLine = OpsDataLine + ";0000"
  ELSE
    OpsDataLine = OpsDataLine + ";" + GetTimeFromTimeIndex(Index_1 - Index_2 + 1)
  END IF

  'store the result
  DKGA_SelectedFlightRecsOrg(LineCounter)= OpsDataLine

  INCR LineCounter

END IF

WEND 'EOF(1)

CLOSE #1

PRINT "finished."

ELSE

  PRINT "finished."

END IF

'at this point records have been read -----
-----

gDKGA_NumberOfRecords = LineCounter-1

'--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---
REDIM gAllFlightsOnThatDay(1 TO gDKGA_NumberOfRecords) AS STRING
'--- REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM ---

DIM DKGA_TempHelpArray(1 TO gDKGA_NumberOfRecords) AS INTEGER

FOR CurrentPos = 1 TO gDKGA_NumberOfRecords

  'copy only filled records
  IF PARSE$(DKGA_SelectedFlightRecsOrg(CurrentPos),";", %FIELD_FlightNumber) <> "" THEN
    gAllFlightsOnThatDay(CurrentPos) = DKGA_SelectedFlightRecsOrg(CurrentPos)
    DKGA_TempHelpArray(CurrentPos) = VAL(PARSE$(gAllFlightsOnThatDay(CurrentPos),";", %FIELD_TimeIndex))
  END IF

NEXT CurrentPos

'Sort the array (TimeIndex/ascending) gAllFlightsOnThatDay() using DKGA_TempHelpArray() as a help array
ARRAY SORT DKGA_TempHelpArray(), TAGARRAY gAllFlightsOnThatDay()

'Fill the GateTime-Matrix with 'empty-default-values'

FOR TimeIndex = 1 TO 288
  FOR GateIndex = 1 TO gNumberOfGates
initialization
    gGateTime_Matrix(TimeIndex, GateIndex) = %NoFlightValue
  flights
  NEXT GateIndex
NEXT TimeIndex

'5-min-intervals (00:00 - 23:55)
'gNumberOfGates is set during program
'later the cell will be replaced by index to

END SUB 'OPTI_Initialize()

'-----

SUB OPTI_InitializeOptiParamFromFile()

LOCAL FileHandle AS INTEGER
LOCAL ParameterLine AS STRING
LOCAL FoundVariables AS INTEGER : FoundVariables = 0
LOCAL ExpectedVariables AS INTEGER : ExpectedVariables = 3

```

```

FileHandle = FREEFILE

OPEN FILE_OPTIPARAMETERS FOR INPUT AS #FileHandle

WHILE NOT EOF(FileHandle)
    LINE INPUT #FileHandle, ParameterLine
    IF LEFT$(ParameterLine, 2) <> "//" AND LEN(TRIM$(ParameterLine)) > 0 THEN

        SELECT CASE UCASE$(PARSE$(ParameterLine, ";", 1))

            CASE "MINUTESATGATEMINIMUM"
                gDKGA_MinutesAtGateMINIMUM = VAL(PARSE$(ParameterLine, ";", 2))
                INCR FoundVariables

            CASE "MINUTESATGATEMAXIMUM"
                gDKGA_MinutesAtGateMAXIMUM = VAL(PARSE$(ParameterLine, ";", 2))
                INCR FoundVariables

            CASE "BUFFERTIMEATGATE"
                gBufferIntervals = INT(VAL(PARSE$(ParameterLine, ";", 2)) / 5)
                INCR FoundVariables

            CASE ELSE
                ConsoleMessageBox "An unknown variable has been found in FILE_OPTIPARAMETERS:\n" + _
                    ParameterLine, %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0

        END SELECT

        END IF
    WEND 'EOF(FileHandle)

IF FoundVariables <> ExpectedVariables THEN
    ConsoleMessageBox "Number of expected variables in FILE_OPTIPARAMETERS: " + STR$(ExpectedVariables), _
        %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
END IF

CLOSE #FileHandle

END SUB 'OPTI_InitializeOptiParamFromFile()

'-----

SUB OPTI_GateTimeMatrix_FillWithFlightPlanOfSingleDay()

    LOCAL GateIndex AS INTEGER
    LOCAL TimeIndex AS INTEGER

    LOCAL CurrentFlightRec AS INTEGER

    'Fills the matrix with 'empty-default-values'

    FOR TimeIndex = 1 TO 288                                '5-min-intervals (00:00 - 23:55)
        FOR GateIndex = 1 TO gNumberOfGates                 'gNumberOfGates is set during program
        initialization
            gGateTime_Matrix(TimeIndex, GateIndex) = %NoFlightValue 'later the cell will be replaced by index to
        flights
        NEXT GateIndex
    NEXT TimeIndex

    'Now fill with flight plan data

    FOR CurrentFlightRec = 1 TO UBOUND(gAllFlightsOnThatDay())

        IF OPTI_GateTimeMatrix_InsertFlight(CurrentFlightRec, _
            GetGateIndexFromGate(PARSE$(gAllFlightsOnThatDay(CurrentFlightRec), ";", %FIELD_Gate_Actual)), _
            GetTimeIndexFromTime(PARSE$(gAllFlightsOnThatDay(CurrentFlightRec), ";", %FIELD_STAG)), _
            VAL(PARSE$(gAllFlightsOnThatDay(CurrentFlightRec), ";", %FIELD_TimeIndex)),
            -
            %FALSE) = %FALSE THEN

            ConsoleMessageBox "Flight could not be inserted!",
                %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
            EXIT FOR
        END IF

    NEXT CurrentFlightRec

END SUB 'OPTI_GateTimeMatrix_FillWithFlightPlanOfSingleDay()

'-----

FUNCTION OPTI_GateTimeMatrix_InsertFlight( BYVAL FlightIndex AS INTEGER, _
    BYVAL Gate AS INTEGER, _
    BYVAL TimeFrom AS INTEGER, _
    BYVAL TimeTo AS INTEGER, _

```



```

        BYVAL OPTI_TrialMode AS INTEGER) AS INTEGER

'OPTI_TrialMode: FALSE => A flight will be inserted / TRUE => it will be TESTED for insert ONLY

LOCAL TimeIndex AS INTEGER
LOCAL LastTimeIndex AS INTEGER

OPTI_GateTimeMatrix_InsertFlight = %FALSE

'check whether it fits
LastTimeIndex = TimeTo + gBufferIntervals

'it is assumed that this is last flight on that day starting
'on that gate, so that no more buffer is needed
IF LastTimeIndex > 288 THEN
    LastTimeIndex = 288
END IF

FOR TimeIndex = TimeFrom TO LastTimeIndex
    IF gGateTime_Matrix(TimeIndex, Gate) <> %NoFlightValue THEN
        'gate is either occupied or blocked with a time gate buffer or blocked with (pax load) block value
        EXIT FUNCTION
    END IF
NEXT TimeIndex

'check for dependency on other gate usage
'the following pairs cannot be used:
'B9:B41 / B19:B20 / E10:E23 / E11:E24 / E12:E25 / E13:E26

IF OPTI_DependendGateIsFree(Gate, TimeFrom) = %FALSE THEN
    OPTI_GateTimeMatrix_InsertFlight = %FALSE
    EXIT FUNCTION
END IF

'gate fits, so insert now the FlightIndex, if not in trial mode
IF OPTI_TrialMode = %FALSE THEN

    FOR TimeIndex = TimeFrom TO TimeTo
        gGateTime_Matrix(TimeIndex, Gate) = FlightIndex
    NEXT TimeIndex

    FOR TimeIndex = TimeTo + 1 TO TimeTo + gBufferIntervals
        gGateTime_Matrix(TimeIndex, Gate) = %BufferValue
    NEXT TimeIndex

END IF

OPTI_GateTimeMatrix_InsertFlight = %TRUE

END FUNCTION 'OPTI_GateTimeMatrix_InsertFlight()

-----

FUNCTION OPTI_GateTimeMatrix_RemoveFlight(BYVAL FlightIndex AS INTEGER, BYVAL Gate AS INTEGER, BYVAL TimeFrom AS
INTEGER, BYVAL TimeTo AS INTEGER) AS INTEGER

LOCAL TimeIndex AS INTEGER

OPTI_GateTimeMatrix_RemoveFlight = %FALSE

'check whether it exists in entire period
FOR TimeIndex = TimeFrom TO TimeTo
    IF gGateTime_Matrix(TimeIndex, Gate) <> FlightIndex THEN
        'gate is occupied by other flight or already empty
        ConsoleMessageBox "Gate occupied by OTHER flight or EMPTY!" + _
            "\n\nGate: " + STR$(Gate) + "\nTimeIndex: " + STR$(TimeIndex) + _
            "\nTimeFrom: " + STR$(TimeFrom) + "\nTimeTo: " + STR$(TimeTo) + _
            "\nFlight: " + STR$(FlightIndex) + _
            "\n\nFlightIndex: gGateTime_Matrix(TimeIndex, Gate)" +
STR$(gGateTime_Matrix(TimeIndex, Gate)), _
            %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
        EXIT FUNCTION
    END IF
NEXT TimeIndex

'check whether the calling function perhaps handed over wrong parameter values, or the flight schedule has been
corrupted
IF gGateTime_Matrix(TimeTo+1, Gate) = FlightIndex THEN
    ConsoleMessageBox "Detected that gate is longer occupied by flight than requested to remove!" + _
        "\n\nGate: " + STR$(Gate) + "\nTime: " + STR$(TimeTo+1) + "\nFlight: " +
STR$(FlightIndex), _
        %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
    EXIT FUNCTION
END IF

'gate is entirely occupied by FlightIndex, so set now to empty

```

```

FOR TimeIndex = TimeFrom TO TimeTo + gBufferIntervals
  gGateTime_Matrix(TimeIndex, Gate) = %NoFlightValue
NEXT TimeIndex

OPTI_GateTimeMatrix_RemoveFlight = %TRUE

END FUNCTION 'OPTI_GateTimeMatrix_RemoveFlight()

'-----

SUB OPTI_GateTimeMatrix_Show(BYVAL Gate AS INTEGER, BYVAL TimeFrom AS INTEGER, BYVAL TimeTo AS INTEGER)

LOCAL TimeIndex AS INTEGER
LOCAL LastShownFlightIndex AS INTEGER

IF TimeTo < TimeFrom THEN
  ConsoleMessageBox "This Message should not occur!\n\nTimeTo < TimeFrom.",
%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
  EXIT SUB
END IF

LastShownFlightIndex = gGateTime_Matrix(TimeFrom, Gate)
PRINT
PRINT "|";
PRINT USING$("####", LastShownFlightIndex);

FOR TimeIndex = TimeFrom+1 TO TimeTo

  IF gGateTime_Matrix(TimeIndex, Gate) <> LastShownFlightIndex THEN
    PRINT "|";
    LastShownFlightIndex = gGateTime_Matrix(TimeIndex, Gate)
  END IF
  PRINT USING$("####", gGateTime_Matrix(TimeIndex, Gate));

NEXT TimeIndex

END SUB 'OPTI_GateTimeMatrix_Show()

'-----

SUB OPTI_GateTimeMatrix_DEBUG()

LOCAL i AS INTEGER
LOCAL TempGateName AS STRING

'DEBUG: write array into file to check ok
PRINT "Writing GateTimeMatrix into debug file...";
OPEN PATH_APPLICATION+$FILE_DEBUG FOR OUTPUT AS #99

'headline with gates
PRINT #99, "----: ";
FOR i = 1 TO gNumberOfGates
  TempGateName = OPTI_Geno2Pheno_Gate(STR$(i))
  PRINT #99, SPACE$(4-LEN(TempGateName)) + TempGateName + "|";
NEXT i

PRINT #99, "-"

FOR DEBUG_COUNTER_1 = 1 TO 288
  PRINT #99, USING$("###",DEBUG_COUNTER_1) + ": ";
  FOR DEBUG_COUNTER_2 = 1 TO gNumberOfGates
    PRINT #99, USING$("###",gGateTime_Matrix(DEBUG_COUNTER_1, DEBUG_COUNTER_2))+ "|";
  NEXT DEBUG_COUNTER_2
  PRINT #99, " "
NEXT DEBUG_COUNTER_1
CLOSE #99
PRINT "finished."

END SUB 'OPTI_GateTimeMatrix_DEBUG()

'-----

FUNCTION OPTI_Pheno2Geno_Gate(BYVAL PhenoGate AS STRING) AS STRING

'this function accepts Gates as input and delivers an integer number (STRING) as output

LOCAL GatesInfraFieldNo_Index AS INTEGER : GatesInfraFieldNo_Index = 1
LOCAL GatesInfraFieldNo_GateName AS INTEGER : GatesInfraFieldNo_GateName = 2
LOCAL i AS LONG

IF TRIM$(PhenoGate) <> "" THEN

  FOR i = 1 TO UBOUND(gDKGA_GatesInfra())

    IF TRIM$(PARSE$(gDKGA_GatesInfra(i), ";", GatesInfraFieldNo_GateName)) = TRIM$(PhenoGate) THEN
      OPTI_Pheno2Geno_Gate = PARSE$(gDKGA_GatesInfra(i), ";", GatesInfraFieldNo_Index)
      EXIT FUNCTION
    END IF
  END IF

```

```

NEXT i

ELSE 'no gate info / gate is not filled

OPTI_Pheno2Geno_Gate = "1"
ConsoleMessageBox "This Message should not occur!\n\nA gate (NAME) is not filled ==> set to '1.'" + _
    "\n\nVALUE:" + PhenoGate + ".", %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

EXIT FUNCTION

END IF

ConsoleMessageBox "This Message should not occur!\n\nA gate (NAME) could not be found during OPTI run." + _
    "\n\nVALUE:" + PhenoGate + ".", %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

END FUNCTION 'OPTI_Pheno2Geno_Gate()

'-----

FUNCTION OPTI_Geno2Pheno_Gate(BYVAL GenoGate AS STRING) AS STRING

'this function accepts Integer Values (STRING) as input and delivers the appropriate gate as output
'it is used to convert the integer (generated by Evolver) of a day's flight plan to real gate names
'to be used for fitness determination

LOCAL GatesInfraFieldNo_Index AS INTEGER : GatesInfraFieldNo_Index = 1
LOCAL GatesInfraFieldNo_GateName AS INTEGER : GatesInfraFieldNo_GateName = 2
LOCAL i AS LONG

IF TRIM$(GenoGate) <> "" THEN

FOR i = 1 TO UBOUND(gDKGA_GatesInfra())

IF TRIM$(PARSE$(gDKGA_GatesInfra(i), ";", GatesInfraFieldNo_Index)) = TRIM$(GenoGate) THEN
OPTI_Geno2Pheno_Gate = PARSE$(gDKGA_GatesInfra(i), ";", GatesInfraFieldNo_GateName)
EXIT FUNCTION
END IF

NEXT i

ELSE

'no gate info / gate is not filled
OPTI_Geno2Pheno_Gate = "A1"

ConsoleMessageBox "This Message should not occur!\n\nA gate (Geno2Pheno) could not be found ==> set to 'A1'."
+ _
    "\n\nVALUE:" + GenoGate + ".", %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

EXIT FUNCTION

END IF

ConsoleMessageBox "This Message should not occur!\n\nA gate (INTEGER) could not be found during OPTI run." + _
    "\n\nVALUE:" + GenoGate + ".", %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

END FUNCTION 'OPTI_Geno2Pheno_Gate()

'-----

FUNCTION GetTimeFromMinutes(BYVAL MinuteValue AS INTEGER) AS STRING

LOCAL TempTime AS STRING
LOCAL TempHour AS INTEGER
LOCAL TempMinutes AS INTEGER

TempHour = INT(MinuteValue/60)
TempMinutes = MinuteValue - (TempHour*60)

SELECT CASE TempHour
CASE 0
TempTime = "00"
CASE 1 TO 9
TempTime = "0" + TRIM$(STR$(TempHour))
CASE ELSE
TempTime = TRIM$(STR$(TempHour))
END SELECT

SELECT CASE TempMinutes
CASE 0
TempTime = TempTime + "00"
CASE 1 TO 9
TempTime = TempTime + "0" + TRIM$(STR$(TempMinutes))
CASE ELSE
TempTime = TempTime + TRIM$(STR$(TempMinutes))
END SELECT

GetTimeFromMinutes = TempTime

END FUNCTION 'GetTimeFromMinutes(BYVAL MinuteValue AS INTEGER) AS STRING

```

```

-----
FUNCTION GetTimeIndexFromTime(BYVAL DateTime AS STRING) AS INTEGER
    'DateTime in Format: HHMM
    LOCAL TimeIndex AS INTEGER

    'Calculation of TimeIndex: (hours*12) + INT(minutes/5) + 1

    TimeIndex = VAL(LEFT$(DateTime, 2)) * 12 'hours
    TimeIndex = TimeIndex + INT(VAL(RIGHT$(DateTime, 2)) / 5)
    TimeIndex = TimeIndex + 1

    GetTimeIndexFromTime = TimeIndex
END FUNCTION 'GetTimeIndexFromTime()
-----

FUNCTION GetTimeFromTimeIndex(BYVAL TimeIndex AS INTEGER) AS STRING

    LOCAL TempHours AS INTEGER
    LOCAL TempMinutes AS INTEGER
    LOCAL TempTime AS STRING

    TempHours = INT((TimeIndex-1)/12)
    TempMinutes = ((TimeIndex-1) - (TempHours * 12)) * 5

    SELECT CASE TempHours
        CASE 0
            TempTime = "00"
        CASE 1 TO 9
            TempTime = "0" + TRIM$(STR$(TempHours))
        CASE ELSE
            TempTime = TRIM$(STR$(TempHours))
    END SELECT

    SELECT CASE TempMinutes
        CASE 0
            TempTime = TempTime + "00"
        CASE 1 TO 9
            TempTime = TempTime + "0" + TRIM$(STR$(TempMinutes))
        CASE ELSE
            TempTime = TempTime + TRIM$(STR$(TempMinutes))
    END SELECT

    GetTimeFromTimeIndex = TempTime
END FUNCTION 'GetTimeFromTimeIndex()
-----

FUNCTION GetGateIndexFromGate(BYVAL Gate AS STRING) AS INTEGER

    LOCAL i AS INTEGER

    IF Gate <> "" AND Gate <> "N" THEN      ' "N" is the init/default value from Data Cleansing

        FOR i = 1 TO UBOUND(gDKGA_GatesInfra())
            IF TRIM$(PARSE$(gDKGA_GatesInfra(i), ";", 2)) = Gate THEN
                GetGateIndexFromGate = VAL(TRIM$(PARSE$(gDKGA_GatesInfra(i), ";", 1)))
                EXIT FUNCTION
            END IF
        NEXT i

    ELSE

        ConsoleMessageBox "This should not occur (GetGateIndexFromGate): Gate EMPTY!",
        %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
        EXIT FUNCTION

    END IF

    ConsoleMessageBox "This should not occur (GetGateIndexFromGate): Gate not found " + Gate ,
    %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
END FUNCTION 'GetGateIndexFromGate()
-----

FUNCTION GetRetailAreaIndexFromRetailArea(BYVAL RetailArea AS STRING) AS INTEGER

    LOCAL i AS INTEGER

    IF RetailArea <> "" THEN

```

```

FOR i = 1 TO UBOUND(gaRetailArea())
  IF TRIM$(PARSE$(gaRetailArea(i), ";", 1)) = RetailArea THEN
    GetRetailAreaIndexFromRetailArea = i
    EXIT FUNCTION
  END IF
NEXT i

ELSE

  ConsoleMessageBox "This should not occur (GetRetailAreaIndexFromRetailArea): GetRetailArea EMPTY!",
%OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
  EXIT FUNCTION

END IF

  ConsoleMessageBox "This should not occur (GetRetailAreaIndexFromRetailArea): Retailarea not found " + RetailArea ,
%OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
  GetRetailAreaIndexFromRetailArea = 0

END FUNCTION 'GetRetailAreaIndexFromRetailArea()

'-----

FUNCTION GetRetailAreaIndexFromGateIndex(BYVAL GateIndex AS INTEGER) AS INTEGER

  LOCAL i AS INTEGER
  LOCAL j AS INTEGER

  LOCAL GateName AS STRING

  GetRetailAreaIndexFromGateIndex = 0

  GateName = PARSE$(gDKGA_GatesInfra(GateIndex), ";", 2)

  IF GateIndex <> 0 THEN

    FOR i = 1 TO UBOUND(gaRetailArea())-1

      FOR j = 2 TO PARSECOUNT(gaRetailArea(i), ";") ' "2" because of structure of RetailAreaDef:
"R2;B1;B2;B3;B4;B5;B6;B7;B8;B9"

        IF PARSE$(gaRetailArea(i), ";", j) = GateName THEN
          GetRetailAreaIndexFromGateIndex = i
          EXIT FUNCTION
        END IF
      NEXT j

    NEXT i

  ELSE

    ConsoleMessageBox "This should not occur (GetRetailAreaIndexFromGateIndex): GateIndex = 0!",
%OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
    EXIT FUNCTION

  END IF

END FUNCTION 'GetRetailAreaIndexFromGateIndex()

'-----

FUNCTION OPTI_Determine_AvailableGatesInInterval(BYVAL SpecificTimeInterval AS INTEGER) AS STRING

  LOCAL GateIndex AS INTEGER
  LOCAL TempString AS STRING : TempString = ""

  RESET gRetailAreaNumberOfFreeGates()

  FOR GateIndex = 1 TO gNumberOfGates
    IF gGateTime_Matrix(SpecificTimeInterval, GateIndex) = %NoFlightValue THEN
      'gate is not occupied, but...
      'may not be available, because a dependend gate is already occupied

      IF OPTI_DependendGateIsFree(GateIndex, SpecificTimeInterval) = %TRUE THEN

        TempString = TempString + TRIM$(STR$(GateIndex)) + ";"

        'update the number of free gates per retail area (for later use)
        INCR gRetailAreaNumberOfFreeGates(GetRetailAreaIndexFromGateIndex(GateIndex))

      END IF

    END IF
  NEXT GateIndex

  IF RIGHT$(TempString, 1) = ";" THEN
    TempString = LEFT$(TempString, LEN(TempString)-1) ' remove the last semicolon
  END IF

  OPTI_Determine_AvailableGatesInInterval = TempString

```

```

END FUNCTION 'OPTI_Determine_AvailableGatesInInterval()

'-----

FUNCTION OPTI_Determine_FlightsToBeAllocatedInTimeInterval(BYVAL SpecificTimeInterval AS INTEGER) AS STRING

    LOCAL i AS INTEGER
    LOCAL TempString AS STRING : TempString = ""

    FOR i = 1 TO UBOUND(gAllFlightsOnThatDay())
        IF GetTimeIndexFromTime(PARSE$(gAllFlightsOnThatDay(i), ";", %FIELD_STAG)) = SpecificTimeInterval THEN
            TempString = TempString + TRIM$(STR$(i)) + ";"
        END IF
    NEXT i

    TempString = LEFT$(TempString, LEN(TempString)-1) ' remove the last semicolon
    OPTI_Determine_FlightsToBeAllocatedInTimeInterval = TempString

END FUNCTION 'OPTI_Determine_FlightsToBeAllocatedInTimeInterval()

'-----

FUNCTION OPTI_Determine_EligibleGatesForFlight(BYVAL FlightIndex AS INTEGER, BYVAL OPTI_GatesAvailable AS STRING) AS STRING

    'returns String of GateIndices, e.g. "1;22;34;45;22"

    LOCAL TrialGateIndex AS INTEGER
    LOCAL TempString AS STRING : TempString = ""

    IF FlightIndex <> 0 THEN

        FOR TrialGateIndex = 1 TO PARSECOUNT(OPTI_GatesAvailable, ";")
            IF OPTI_IsValidGate(FlightIndex, PARSE$(OPTI_GatesAvailable, ";", TrialGateIndex)) = %TRUE THEN
                TempString = TempString + PARSE$(OPTI_GatesAvailable, ";", TrialGateIndex) + ";"
            END IF
        NEXT TrialGateIndex

        IF RIGHT$(TempString,1) = ";" THEN
            TempString = LEFT$(TempString, LEN(TempString)-1) ' remove the last semicolon
        END IF

        OPTI_Determine_EligibleGatesForFlight = TempString

    ELSE
        ConsoleMessageBox "FlightIndex = 0 !!!", %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
    END IF

END FUNCTION 'OPTI_Determine_EligibleGatesForFlight()

'-----

FUNCTION OPTI_IsValidGate( BYVAL FlightIndex AS INTEGER, _
    BYVAL TrialGateIndex AS STRING) AS INTEGER

    LOCAL TempStandOpti AS STRING
    LOCAL lGateFound AS INTEGER
    LOCAL j AS INTEGER
    LOCAL FieldGateMaxWSC AS INTEGER : FieldGateMaxWSC = 7

    '-----

    OPTI_IsValidGate = %TRUE

    TempStandOpti = PARSE$(gDKGA_GatesInfra(VAL(TrialGateIndex)), ";", 2)

    '-----
    '--- check 1: aircraft size vs. STAND size ---
    '-----

    'The higher the WSC the smaller the A/C must be (the larger the A/C the smaller the WSC: e.g. B744 = WSC 2)

    'normalize gate names (e.g. 'D11A' --> 'D11')
    IF RIGHT$(TRIM$(TempStandOpti),1) = "A" THEN
        TempStandOpti = TRIM$(TempStandOpti)
        TempStandOpti = TRIM$(LEFT$(TempStandOpti, LEN(TempStandOpti)-1 ))
    END IF

    'check only if there is a value for Stand_Opti. in case not: incr a counter

```

```

IF TRIM$(TempStandOpti) <> "" THEN

'check for size only in case of a contact gate NOT in case of a remote stand
IF IsRemoteStand(TempStandOpti)= %FALSE THEN 'means it is a contact gate

    lGateFound = %FALSE
    FOR j = 1 TO gNumberOfGates

        'look for matching CONTACT gate (STAND = GATE)

        IF TempStandOpti = TRIM$(PARSE$(gDKGA_GatesInfra(j), ";", 2)) THEN

            'check for A/C WSC of flight:
            'if GateInfra.WSC > FlightRec.Gate.WSC

                IF TRIM$(PARSE$(gAllFlightsOnThatDay(FlightIndex), ";", %FIELD_ACType)) = "" THEN
                    ConsoleMessageBox "This should not occur: EMPTY A/C!" + _
                        "\nFlight      : " + PARSE$(gAllFlightsOnThatDay(FlightIndex), ";",
%FIELD_FlightNumber) + _
                        "\nFlightIndex : " + STR$(FlightIndex), _
%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
                    OPTI_IsValidGate = %FALSE
                    EXIT FUNCTION
                ELSE
                    IF VAL(TRIM$(PARSE$(gDKGA_GatesInfra(j), ";", FieldGateMaxWSC))) >
VAL(GetWingSpanCode(PARSE$(gAllFlightsOnThatDay(FlightIndex), ";", %FIELD_ACType))) THEN
                        'gate is too small i.e. A/C is too large
                        OPTI_IsValidGate = %FALSE
                        EXIT FUNCTION
                    END IF
                END IF

                j = gNumberOfGates 'EXIT FOR
                lGateFound = %TRUE

            END IF 'so try next gate/stand

        NEXT j

        IF lGateFound = %FALSE THEN
            ConsoleMessageBox "This should not occur: Stand not found " + TempStandOpti ,
%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
            END IF

        END IF 'IsRemoteGate = %FALSE

    ELSE
        'an empty value for stand results in an invalid solution
        ConsoleMessageBox "This should not occur: EMPTY GATE! No Gate handed over to function 'IsValidGate()',
%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
        OPTI_IsValidGate = %FALSE
        END IF

        'at this position OPTI_IsValidGate = %TRUE (function would have been EXITed before, if not), so continue checks...

        '-----
        '--- check 2: airline alliances          ---
        '-----

        IF glSecondTryWithoutAllianceCompliance = %FALSE THEN

            SELECT CASE UCASE$(GetAlliance(LEFT$(PARSE$(gAllFlightsOnThatDay(FlightIndex), ";", %FIELD_FlightNumber),2)))

                CASE "STARALLIANCE"

                    'look for a 'Star Alliance' CKI hall
                    IF TALLY(GetCKIHallFromGate(TempStandOpti), ANY "ABC") = 0 THEN 'no appropriate CKI hall found
                        OPTI_IsValidGate = %FALSE
                        EXIT FUNCTION
                    END IF

                CASE "SKYTEAM"

                    'look for a 'SkyTeam' CKI hall
                    IF TALLY(GetCKIHallFromGate(TempStandOpti), ANY "DE") = 0 THEN 'no appropriate CKI hall found
                        OPTI_IsValidGate = %FALSE
                        EXIT FUNCTION
                    END IF

                CASE "ONeworld"

                    'look for a 'oneworld' CKI hall
                    IF TALLY(GetCKIHallFromGate(TempStandOpti), ANY "DE") = 0 THEN 'no appropriate CKI hall found
                        OPTI_IsValidGate = %FALSE
                        EXIT FUNCTION
                    END IF

                CASE ELSE 'flight does not belong to an alliance, so it does not matter which CKI hall (FOR ACADEMIC
SIMPLIFICATION)

                    OPTI_IsValidGate = %TRUE

            END SELECT

```

```

END IF 'glSecondTryWithoutAllianceCompliance = %FALSE
END FUNCTION 'OPTI_IsValidGate()

'-----

FUNCTION OPTI_Determine_EligibleRetailAreasForFlight(BYVAL SetOfGateIndices AS STRING) AS STRING

LOCAL i AS INTEGER
LOCAL j AS INTEGER

LOCAL TempGate AS STRING
LOCAL TempRetailArea AS STRING
LOCAL TempSetOfRetailAreaIndices AS STRING

LOCAL NoOfGatesInSet AS INTEGER

IF TALLY(SetOfGateIndices, ";") = 0 AND LEN(SetOfGateIndices) > 0 THEN
    NoOfGatesInSet = 1
    SetOfGateIndices = SetOfGateIndices + ";"
ELSE
    NoOfGatesInSet = PARSECOUNT(SetOfGateIndices, ";")
END IF

'for every gate...

' PRINT #99, " "
' PRINT #99, "SetOfGateIndices: "; SetOfGateIndices
' PRINT #99, "TempGate: ";
FOR i = 1 TO NoOfGatesInSet

    TempGate = PARSE$(gDKGA_GatesInfra(VAL(PARSE$(SetOfGateIndices, ";", i)),";",2))

    PRINT #99, TempGate + "-";

    '...determine RetailArea from a Gate given
    FOR j = 1 TO UBOUND(gDKGA_GatesInfra())
        IF TALLY(gaRetailArea(j), TempGate) > 0 THEN
            TempRetailArea =
TRIM$(STR$(GetRetailAreaIndexFromRetailArea(LEFT$(gaRetailArea(j),2))))
            EXIT FOR
        END IF
    NEXT j

    IF TempRetailArea = "" THEN
        ConsoleMessageBox "This should not occur: EMPTY RETAIL AREA has been returned!",
%OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0
    END IF

    'if that retail area is not yet in list of eligible retail areas for that flight, add it
    IF TALLY(TempSetOfRetailAreaIndices, TempRetailArea) = 0 THEN
        TempSetOfRetailAreaIndices = TempSetOfRetailAreaIndices + TempRetailArea + ";"
    END IF

NEXT i
PRINT #99, " "

IF RIGHT$(TempSetOfRetailAreaIndices, 1) = ";" THEN
    TempSetOfRetailAreaIndices = LEFT$(TempSetOfRetailAreaIndices, LEN(TempSetOfRetailAreaIndices)-1) ' remove the
last semicolon
END IF

OPTI_Determine_EligibleRetailAreasForFlight = TempSetOfRetailAreaIndices
END FUNCTION 'OPTI_Determine_EligibleRetailAreasForFlight()

'-----

FUNCTION OPTI_Determine_RevenueForFlightInSpecificRetailArea(BYVAL FlightIndex AS STRING, BYVAL SetOfRetailAreaIndices
AS STRING) AS STRING

LOCAL i AS INTEGER
LOCAL TempRevenue AS DOUBLE
LOCAL TempSetOfRevenues AS STRING
LOCAL TempRetailArea AS STRING
LOCAL TempRetailAreaFactor AS STRING
LOCAL FlightPaxDFfactor AS STRING

LOCAL NoOfRetailAreasInSet AS INTEGER

IF TALLY(SetOfRetailAreaIndices, ";") = 0 AND LEN(SetOfRetailAreaIndices) > 0 THEN
    NoOfRetailAreasInSet = 1
    SetOfRetailAreaIndices = SetOfRetailAreaIndices + ";"
ELSE
    NoOfRetailAreasInSet = PARSECOUNT(SetOfRetailAreaIndices, ";")
END IF

FlightPaxDFfactor = PARSE$(gAllFlightsOnThatDay(VAL(FlightIndex)), ";", %FIELD_FlightPAXDFfactor)
REPLACE " ," WITH "." IN FlightPaxDFfactor 'convert FlightPaxDFfactor into a decimal value format

```



```

"121,6")
'just in case a comma instead of decimal point is used ( "121,6" -->

FOR i = 1 TO NoOfRetailAreasInSet

TempRetailArea = PARSE$(gaRetailArea( VAL(PARSE$(SetOfRetailAreaIndices, ";", i)) ), ";", 1)
TempRetailAreaFactor = GetRetailFactor(TempRetailArea, gaRF() )

'formula:
' retail revenue per flight = RevenuePerPax * NumberOfPax * (FlightPaxDFfactor/100)
* DF->RetailFactor * RetailAreaFactor = gRevPerPax * %FIELD_PAX_Actual * (%FIELD_FlightPAXDFfactor /
100) * FILE_DF_RETAIL_FACTOR * %FIELD_RetailAreaFactorActual = 6,5 * 233 * (121,6/100)
* 3,4 * 2,7 = ...

IF VAL(TempRetailAreaFactor) <> 0 THEN

TempRevenue = ROUND ( _
gRevPerPax * _
VAL(PARSE$(gAllFlightsOnThatDay(VAL(FlightIndex)), ";", %FIELD_PAX_Actual) ) *
VAL(FlightPaxDFfactor) / 100 * _
gDF_RetailFactor * _
VAL(TempRetailAreaFactor) ,2)

IF TempRevenue = 0 THEN
ConsoleMessageBox "TempRevenue is ZERO !!!\n" + _
"\nFlightIndex: " + FlightIndex + _
"\nPAX actual: " + PARSE$(gAllFlightsOnThatDay(VAL(FlightIndex)), ";",
%FIELD_PAX_Actual) + _
"\nFlightPaxDFfactor: " + FlightPaxDFfactor + _
"\ngRevPerPax: " + STR$(gRevPerPax) + _
"\ngDF_RetailFactor: " + STR$(gDF_RetailFactor) + _
"\nTempRetailAreaFactor: " + TempRetailAreaFactor, _
%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
END IF

ELSE

ConsoleMessageBox "This Message should not occur!\n\nRetail Revenue for a flight could not be
calculated!." + _
"\n\nFlight: " + PARSE$(gAllFlightsOnThatDay(VAL(FlightIndex)), ";",
%FIELD_FlightNumber) + ".", %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

TempRevenue = 0

END IF

TempSetOfRevenues = TempSetOfRevenues + TRIM$(STR$(TempRevenue)) + ";"

NEXT i

IF RIGHT$(TempSetOfRevenues, 1) = ";" THEN
TempSetOfRevenues = LEFT$(TempSetOfRevenues, LEN(TempSetOfRevenues)-1) ' remove the last semicolon
END IF

OPTI_Determine_RevenueForFlightInSpecificRetailArea = TempSetOfRevenues
END FUNCTION 'OPTI_Determine_RevenueForFlightInSpecificRetailArea()

'-----
FUNCTION OPTI_Determine_RevenueForFlightAtSpecificGates(BYVAL FlightIndex AS STRING, BYVAL SetOfGateIndices AS STRING)
AS STRING

LOCAL i AS INTEGER
LOCAL TempGateRevenues AS STRING : TempGateRevenues = ""

FOR i = 1 TO PARSECOUNT(SetOfGateIndices, ";")

TempGateRevenues = TempGateRevenues + _
OPTI_Determine_RevenueForFlightInSpecificRetailArea( _
FlightIndex, _
TRIM$(STR$(GetRetailAreaIndexFromGateIndex(VAL(PARSE$(SetOfGateIndices, ";", i)) ) ) ) )
) + ";"

IF GetRetailAreaIndexFromGateIndex(VAL(PARSE$(SetOfGateIndices, ";", i) ) ) = 0 THEN
ConsoleMessageBox "GetRetailAreaIndexFromGateIndex = 0 !!!" + _
"\nSetOfGateIndices: " + SetOfGateIndices,
%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
END IF

NEXT i

IF RIGHT$(TempGateRevenues, 1) = ";" THEN
TempGateRevenues = LEFT$(TempGateRevenues, LEN(TempGateRevenues)-1) ' remove the last semicolon
END IF

```

```

OPTI_Determine_RevenueForFlightAtSpecificGates = TempGateRevenues
END FUNCTION 'OPTI_Determine_RevenueForFlightAtSpecificGates
'-----

SUB OPTI_Show_ProgressWindow()

    LOCAL hBmp AS LONG

    LOCAL h, w, hGW AS LONG

    h = 371
    w = 755

    GRAPHIC WINDOW "Allocating Gates...", LocOfCol(22), LocOfRow(5), w, h TO hGW
    gVarNumber_3_TimerFunc = hGW

    GRAPHIC ATTACH hGW, 0&, REDRAW
    GRAPHIC COLOR RGB(0,0,0), RGB(255,255,255)
    GRAPHIC CLEAR

    GRAPHIC BITMAP LOAD PATH_APPLICATION+"OptiRun_Progress.bmp", w, h TO hBmp
    GRAPHIC COPY hBmp, 0 TO (1, 1)
    GRAPHIC REDRAW

END SUB 'OPTI_Show_ProgressWindow()
'-----

SUB OPTI_NoShow_ProgressWindow()

    GRAPHIC BITMAP END
    GRAPHIC WINDOW END

END SUB 'OPTI_NoShow_ProgressWindow()
'-----

FUNCTION OPTI_GateChosen(BYVAL CandidateRetailAreaIndex AS INTEGER, BYVAL CurrentFlight AS INTEGER, BYVAL TimeFrom AS
INTEGER, BYREF OPTI_GateSet_PerFlight() AS STRING) AS INTEGER

    'tries to assign a single flight to a gate within all possible retail areas

    LOCAL i AS INTEGER
    LOCAL j AS INTEGER
    LOCAL TimeTo AS INTEGER
    LOCAL GroundTime AS INTEGER
    LOCAL FlightIndex AS INTEGER
    LOCAL TestGateIndex AS INTEGER

    OPTI_GateChosen = 0
    FlightIndex = VAL(OPTI_GateSet_PerFlight(CurrentFlight, 1))

    IF FlightIndex = 363 THEN
        DEBUG_PRINT = %TRUE
    END IF

    'assure that there is a minimum ground time (at gate) in case the
    'flight plan has a value less than a feasible minimum value

    IF VAL(PARSE$(gAllFlightsOnThatDay(FlightIndex), ";", %FIELD_StdGroundTime)) < gDKGA_MinutesAtGateMINIMUM THEN
        GroundTime = INT(gDKGA_MinutesAtGateMINIMUM / 5)
    ELSE
        GroundTime = INT(VAL(PARSE$(gAllFlightsOnThatDay(FlightIndex), ";", %FIELD_StdGroundTime)) / 5)
    END IF

    'then check whether ground time exceeds max value from opti parameter file
    'and in case it does, take the maximum allowed value

    TimeTo = TimeFrom + MIN( GroundTime , INT(gDKGA_MinutesAtGateMAXIMUM/5) ) - 1

    'for each gate in Retail Area try an insert of flight into Matrix
    ' startindex=2 because of structure in RetailAreaDef-File: e.g. "R4;C1;C2;C4;C5;C6;C7;C8;C9;C11;C13;C21;C22"

    'search within eligible gates in that retail area only !!!
    FOR j = 1 TO PARSECOUNT(OPTI_GateSet_PerFlight(CurrentFlight, 2), ";")

        'try next gate in retail area
        FOR i = 2 TO PARSECOUNT(gaRetailArea(CandidateRetailAreaIndex), ";")

            TestGateIndex = GetGateIndexFromGate(PARSE$(gaRetailArea(CandidateRetailAreaIndex), ";", i))

```

```

'if within eligible gates...
IF TestGateIndex = VAL(PARSE$(OPTI_GateSet_PerFlight(CurrentFlight, 2), ";", j)) THEN

    'try an insert...

    IF OPTI_GateTimeMatrix_InsertFlight(FlightIndex, TestGateIndex, TimeFrom, TimeTo, %FALSE) = %TRUE THEN

        OPTI_GateChosen = TestGateIndex

        'Transaction handling
        INCR gGateTimeMatrix_Transaction_Stack_Counter
        gGateTimeMatrix_Transaction_Stack(gGateTimeMatrix_Transaction_Stack_Counter, 1) = FlightIndex
        gGateTimeMatrix_Transaction_Stack(gGateTimeMatrix_Transaction_Stack_Counter, 2) = TestGateIndex
        gGateTimeMatrix_Transaction_Stack(gGateTimeMatrix_Transaction_Stack_Counter, 3) = TimeFrom
        gGateTimeMatrix_Transaction_Stack(gGateTimeMatrix_Transaction_Stack_Counter, 4) = TimeTo

        EXIT FUNCTION

    ELSE

        '--- trying next eligible gate (---> NEXT j) ---
        EXIT FOR '(i.e. exit i)

    END IF 'InsertFlight

END IF 'TestGateIndex = VAL(PARSE$(OPTI_GateSet_PerFlight(CurrentFlight, 2), ";", j))

NEXT i

NEXT j

END FUNCTION 'OPTI_GateChosen()

'-----

SUB OPTI_GateTimeMatrix_Transaction_Start()

    gGateTimeMatrix_Transaction_Stack_Counter = 0
    RESET gGateTimeMatrix_Transaction_Stack()

END SUB 'OPTI_GateTimeMatrix_Transaction_Start()

'-----

SUB OPTI_GateTimeMatrix_Transaction_Commit()

    'this commit is implemented for INSERTIONS ONLY!

    'actually almost nothing needs to be done in this routine,
    'because flights are inserted for real and no more explicit commit is necessary

    'it is different with commit of deletions and will be programmed when it becomes necessary

    gGateTimeMatrix_Transaction_Stack_Counter = 0
    RESET gGateTimeMatrix_Transaction_Stack()

END SUB 'OPTI_GateTimeMatrix_Transaction_Commit()

'-----

SUB OPTI_GateTimeMatrix_Transaction_Rollback()

    'this rollback is implemented for INSERTIONS ONLY!
    'a rollback will clean up space in matrix (--> free the gates)

    LOCAL i AS INTEGER
    LOCAL DebugString AS STRING

    IF gGateTimeMatrix_Transaction_Stack_Counter = 0 THEN

        ConsoleMessageBox "ROLLBACK not possible because no transactions recorded.\n" + _
            "gGateTimeMatrix_Transaction_Stack_Counter = 0 !!!", _
            %OKONLY+%EXCLAMATIONBOX, "WARNING", %IDI_EXCLAMATION, 0

    ELSE

        'in case the first insert trial was not possible, the gGateTimeMatrix_Transaction_Stack_Counter is 1
        'thus it needs to be 2, in order to rollback that trial

        FOR i = 1 TO gGateTimeMatrix_Transaction_Stack_Counter

            IF OPTI_GateTimeMatrix_RemoveFlight( gGateTimeMatrix_Transaction_Stack(i, 1), _
                gGateTimeMatrix_Transaction_Stack(i, 2), _
                gGateTimeMatrix_Transaction_Stack(i, 3), _
                gGateTimeMatrix_Transaction_Stack(i, 4) ) = %FALSE THEN

```

```

        ConsoleMessageBox "There is a inconsistency problem in ROLLBACK.\nCheck programming code!\n" + _
            "\nFlightIndex   " + STR$(gGateTimeMatrix_Transaction_Stack(i, 1)) + _
            "\nGateIndex     " + STR$(gGateTimeMatrix_Transaction_Stack(i, 2)) + _
            "\nTimeFromIndex  " + STR$(gGateTimeMatrix_Transaction_Stack(i, 3)) + _
            "\nTimeToIndex   " + STR$(gGateTimeMatrix_Transaction_Stack(i, 4)), _
            %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0
    END SUB

NEXT i

END IF

END SUB 'OPTI_GateTimeMatrix_Transaction_Rollback()

'-----

FUNCTION OPTI_DependendGateIsFree(BYVAL Gate AS INTEGER, BYVAL TimeFrom AS INTEGER) AS INTEGER
    LOCAL i AS INTEGER
    LOCAL j AS INTEGER
    LOCAL TestGate_1 AS STRING
    LOCAL TestGate_2 AS STRING
    LOCAL NumberOfDependencies AS INTEGER

    NumberOfDependencies = 36

    DIM DependendGatePair(1 TO NumberOfDependencies) AS STRING

    ' --- no parallel at all ---
    DependendGatePair(1) = "B9;B41"
    DependendGatePair(2) = "B19;B20"
    DependendGatePair(3) = "E10;E23"
    DependendGatePair(4) = "E11;E24"
    DependendGatePair(5) = "E12;E25"
    DependendGatePair(6) = "E13;E26"

    ' --- 20-30 min. buffer ---
    DependendGatePair(7) = "A4;A5"           'A4;A5;30;XX
    DependendGatePair(8) = "B1;B3"           'B1;B3;20;XX
    DependendGatePair(9) = "B1;B4"           'B1;B4;20;XX
    DependendGatePair(10) = "B2;B5"          'B2;B5;20;XX
    DependendGatePair(11) = "B3;B4"          'B3;B4;20;XX
    DependendGatePair(12) = "B6;B7"          'B6;B7;20;XX
    DependendGatePair(13) = "B8;B9"          'B8;B9;20;XX
    DependendGatePair(14) = "D40;D50"        'D40;D50;30;XX
    DependendGatePair(15) = "D41;D51"        'D41;D51;30;XX
    DependendGatePair(16) = "D42;D52"        'D42;D52;30;XX
    DependendGatePair(17) = "D43;D53"        'D43;D53;30;XX
    DependendGatePair(18) = "D44;D54"        'D44;D54;30;XX

    ' --- only in case of both bus gates ---
    DependendGatePair(19) = "A11;A51"
    DependendGatePair(20) = "A12;A52"
    DependendGatePair(21) = "A13;A53"
    DependendGatePair(22) = "A14;A54"
    DependendGatePair(23) = "A15;A55"
    DependendGatePair(24) = "A16;A56"
    DependendGatePair(25) = "A17;A57"
    DependendGatePair(26) = "A18;A58"
    DependendGatePair(27) = "A19;A59"
    DependendGatePair(28) = "A20;A60"
    DependendGatePair(29) = "A21;A61"
    DependendGatePair(30) = "A22;A62"
    DependendGatePair(31) = "A23;A63"
    DependendGatePair(32) = "A25;A65"
    DependendGatePair(33) = "E10;E11"
    DependendGatePair(34) = "E21;E22"
    DependendGatePair(35) = "E23;E24"
    DependendGatePair(36) = "E25;E26"

    ' --- only in case of both bus gates ---
    'A11;A51;30;BB
    'A12;A52;30;BB
    'A13;A53;30;BB
    'A14;A54;30;BB
    'A15;A55;30;BB
    'A16;A56;30;BB
    'A17;A57;30;BB
    'A18;A58;30;BB
    'A19;A59;30;BB
    'A20;A60;30;BB
    'A21;A61;30;BB
    'A22;A62;30;BB
    'A23;A63;30;BB
    'A25;A65;30;BB
    'A51;A11;30;BB
    'A52;A12;30;BB
    'A53;A13;30;BB

```

```

'A54:A14;30;BB
'A55:A15;30;BB
'A56:A16;30;BB
'A57:A17;30;BB
'A58:A18;30;BB
'A59:A19;30;BB
'A60:A20;30;BB
'A61:A21;30;BB
'A62:A22;30;BB
'A63:A23;30;BB
'A65:A25;30;BB
'E10:E11;20;BB
'E11:E10;20;BB
'E21:E22;20;BB
'E22:E21;20;BB
'E23:E24;20;BB
'E24:E23;20;BB
'E25:E26;20;BB
'E26:E25;20;BB

OPTI_DependendGateIsFree = %TRUE
FOR i = 1 TO NumberOfDependencies
  IF OPTI_Geno2Pheno_Gate(TRIM$(STR$(Gate))) = PARSE$(DependendGatePair(i),"",1) OR _
    OPTI_Geno2Pheno_Gate(TRIM$(STR$(Gate))) = PARSE$(DependendGatePair(i),"",2) THEN
    FOR j = 1 TO NumberOfDependencies
      TestGate_1 = PARSE$(DependendGatePair(j),"",1)
      TestGate_2 = PARSE$(DependendGatePair(j),"",2)

      IF OPTI_Geno2Pheno_Gate(TRIM$(STR$(Gate))) = TestGate_1 THEN
        IF gGateTime_Matrix(TimeFrom, VAL(OPTI_Pheno2Geno_Gate(TestGate_2) ) ) <> %NoFlightValue AND _
          gGateTime_Matrix(TimeFrom, VAL(OPTI_Pheno2Geno_Gate(TestGate_2) ) ) <> %BufferValue AND _
          gGateTime_Matrix(TimeFrom, VAL(OPTI_Pheno2Geno_Gate(TestGate_2) ) ) <> %BlockValue THEN
          'dependend gate is occupied
          OPTI_DependendGateIsFree = %FALSE
          EXIT FUNCTION
        END IF
      END IF

      IF OPTI_Geno2Pheno_Gate(TRIM$(STR$(Gate))) = TestGate_2 THEN
        IF gGateTime_Matrix(TimeFrom, VAL(OPTI_Pheno2Geno_Gate(TestGate_1) ) ) <> %NoFlightValue AND _
          gGateTime_Matrix(TimeFrom, VAL(OPTI_Pheno2Geno_Gate(TestGate_1) ) ) <> %BufferValue AND _
          gGateTime_Matrix(TimeFrom, VAL(OPTI_Pheno2Geno_Gate(TestGate_1) ) ) <> %BlockValue THEN
          'dependend gate is occupied
          OPTI_DependendGateIsFree = %FALSE
          EXIT FUNCTION
        END IF
      END IF
    NEXT j
  END IF
NEXT i
END FUNCTION 'OPTI_DependendGateIsFree()

'-----
FUNCTION OPTI_IsValidRACombi(BYVAL TempRAs AS STRING, BYREF OPTI_GateSet_PerFlight() AS STRING) AS INTEGER
  LOCAL i AS INTEGER
  OPTI_IsValidRACombi = %TRUE
  'test each possible retail area
  FOR i = 1 TO (UBOUND(gRetailAreaNumberOfFreeGates())-1)
    'only if the tested retail area is in solution string
    IF TALLY(TempRAs, TRIM$(STR$(i)) ) > 0 THEN
      'count how many occurrences and check whether too many
      IF TALLY(TempRAs, TRIM$(STR$(i)) ) > gRetailAreaNumberOfFreeGates(i) THEN
        OPTI_IsValidRACombi = %FALSE
        EXIT FUNCTION
      END IF
    END IF
  NEXT i
END FUNCTION 'OPTI_IsValidRACombi()

```

```

-----
SUB OPTI_DumpDailyPlanIntoFile(BYVAL CurrentDay AS STRING)

LOCAL i AS LONG
LOCAL DateTimeStamp AS STRING

DateTimeStamp = TIME$
REPLACE ANY ":" WITH "-" IN DateTimeStamp
DateTimeStamp = DATE$ + "_" + DateTimeStamp
OPEN PATH_SCENARIOS + $FILE_OPTIPLAN + "_" + CurrentDay + "_" + DateTimeStamp + ".txt" FOR OUTPUT AS #2

FOR i = 1 TO UBOUND(gAllFlightsOnThatDay())
  PRINT #2, gAllFlightsOnThatDay(i) + ";" + USING$("%0##",i)
NEXT i

CLOSE #2

END SUB 'OPTI_DumpDailyPlanIntoFile()

-----

SUB OPTI_DumpDailyAllocIntoFile(BYVAL CurrentDay AS STRING)

LOCAL i AS LONG
LOCAL j AS LONG
LOCAL DateTimeStamp AS STRING
LOCAL RevPerPax AS CUR
LOCAL DF_RetailFactor AS STRING
LOCAL OpsDataLine AS STRING
LOCAL TempGateName AS STRING

OPEN FILE_DF_RETAIL_FACTOR FOR INPUT AS #2          'File with DF->RetailFactor value

'START: read DF->RetailFactor for later use -----
--

WHILE NOT EOF(2)

  LINE INPUT #2, OpsDataLine

  'the first non-comment-line in file is to be the DF_RetailFactor value
  IF LEFT$(OpsDataLine,2) <> "/" THEN 'no comment line in data file

    REPLACE ", " WITH "." IN OpsDataLine 'just in case a comma instead of decimal point is used ( "3.40" -->
"3.40")

    DF_RetailFactor = OpsDataLine

  END IF

WEND 'EOF(2)

CLOSE #2

'END: read DF->RetailFactor later use -----
--

DateTimeStamp = TIME$
REPLACE ANY ":" WITH "-" IN DateTimeStamp
DateTimeStamp = DATE$ + "_" + DateTimeStamp
OPEN PATH_SCENARIOS + $FILE_OPTIALLOC + "_" + CurrentDay + "_" + DateTimeStamp + ".txt" FOR OUTPUT AS #2

'write header with scenario parameters

'// Minimum GroundTime FOR an A/C AT gate instead OF SGT
'// use full 5 minute values (multiples OF 5) only
'// sample value: 60
'MinutesAtGateMINIMUM;90
'
'// Maximum GroundTime FOR an A/C AT gate instead OF SGT
'// use full 5 minute values (multiples OF 5) only
'// sample value: 120
'MinutesAtGateMAXIMUM;180
'
'// Buffer time between aircraft USING the same gate
'// use full 5 minute values (multiples OF 5) only
'BufferTimeAtGate;20

PRINT #2, "Description of Scenario:"
PRINT #2, "-----"
PRINT #2, " "
PRINT #2, "MinutesAtGateMINIMUM = " + TRIM$(STR$(gDKGA_MinutesAtGateMINIMUM))
PRINT #2, "MinutesAtGateMAXIMUM = " + TRIM$(STR$(gDKGA_MinutesAtGateMAXIMUM))
PRINT #2, "BufferTimeAtGate = " + TRIM$(STR$(gBufferIntervals*5))
PRINT #2, " "
PRINT #2, "DF-Revenue per PAX = " + TRIM$(STR$(gRevPerPax))
PRINT #2, "DF->Retail-Factor = " + DF_RetailFactor

```

```

PRINT #2, " "
PRINT #2, "Retail Area factors:"
FOR i = 1 TO UBOUND(gaRF())
  IF gaRF(i,1) <> "" THEN
    PRINT #2, gaRF(i,1) + " = " + gaRF(i,2)
  END IF
NEXT i
PRINT #2, " "
PRINT #2, "Gate Allocation:"
PRINT #2, "-----"
PRINT #2, " "

'headline with gates
PRINT #2, "----: ";
FOR i = 1 TO gNumberOfGates
  TempGateName = OPTI_Geno2Pheno_Gate(STR$(i))
  PRINT #2, SPACE$(4-LEN(TempGateName)) + TempGateName + "|";
NEXT i

PRINT #2, " "

'allocation in each interval
FOR i = 1 TO 288
  PRINT #2, USING$( "###", i) + ": ";
  FOR j = 1 TO gNumberOfGates
    PRINT #2, USING$( "###", gGateTime_Matrix(i, j)) + "|";
  NEXT j
  PRINT #2, " "
NEXT i

CLOSE #2

END SUB 'OPTI_DumpDailyAllocIntoFile()

'-----

SUB OPTI_Report()

'select file to report on
'test for correct format
'for each record in file (grouped by retail area, day of week)
' actual pax
' revenue actual
' revenue seasonal plan
' revenue opti
' count for invalid entries found

'write result (header & detail)
'header:
' date of report
' dates of data
' opti parameters
'detail:
' =====
' PAX
' =====
'
' MON TUE WED THU FRI SAT SUN WEEK
' R1
' R2
' R3
' R4
' R5
' R6
' R7
'-----
' SUM mon tue wed thu fri sat sun total
'
' =====
' REVENUE ACTUAL
' =====
'
' MON TUE WED THU FRI SAT SUN WEEK
' R1
' R2
' R3
' R4
' R5
' R6
' R7
'-----
' SUM mon tue wed thu fri sat sun total
'
' =====
' REVENUE PLAN
' =====
'
' MON TUE WED THU FRI SAT SUN WEEK
' R1
' R2
' R3
' R4
' R5
' R6
'-----

```

```

' R7
' -----|
' SUM mon tue wed thu fri sat sun total

' =====
' REVENUE OPTI
' =====
' MON TUE WED THU FRI SAT SUN WEEK
' R1
' R2
' R3
' R4
' R5
' R6
' R7
' -----|
' SUM mon tue wed thu fri sat sun total

'AVG TOTAL REVENUE PER DEPARTING PAX
'=====
'ACTUAL = nnn
'PLAN = nnn
'OPTI = nnn

'--- DECLARATIONS -----
LOCAL lFileOpened AS LONG
LOCAL ReportFile AS STRING
LOCAL nFlags AS LONG : nFlags = 0
LOCAL lResult AS LONG : lResult = hConsoleWindow
LOCAL i AS INTEGER
LOCAL j AS INTEGER
DIM ReportDataLine(1 TO 10) AS STRING ' a REDIM will be done below
LOCAL NumberOfRecords AS LONG
LOCAL CurrentRecord AS LONG

LOCAL AllRetailAreas AS STRING
LOCAL lValueFound AS INTEGER

DIM PAX_ACTUAL_RA(1 TO 9, 1 TO 8) AS LONG '1..7 = RAs / 8 = total / 9 = else // 1..7 = MON..SUN / 8 =
total of weekday
DIM PAX_SEASON_RA(1 TO 9, 1 TO 8) AS LONG '1..7 = RAs / 8 = total / 9 = else // 1..7 = MON..SUN / 8 =
total of weekday
DIM PAX_OPTI_RA(1 TO 9, 1 TO 8) AS LONG '1..7 = RAs / 8 = total / 9 = else // 1..7 = MON..SUN / 8 =
total of weekday

DIM Revenue_ACTUAL_RA(1 TO 9, 1 TO 8) AS DOUBLE '1..7 = RAs / 8 = total / 9 = else // 1..7 = MON..SUN / 8 =
total of weekday
DIM Revenue_SEASON_RA(1 TO 9, 1 TO 8) AS DOUBLE '1..7 = RAs / 8 = total / 9 = else // 1..7 = MON..SUN / 8 =
total of weekday
DIM Revenue_OPTI_RA(1 TO 9, 1 TO 8) AS DOUBLE '1..7 = RAs / 8 = total / 9 = else // 1..7 = MON..SUN / 8 =
total of weekday

DIM RA_Counter(1 TO 7) AS LONG

LOCAL SumRevenue AS DOUBLE

'--- ROUTINE -----
CALL LogEntry(FUNCNAME$, "START")

'--- choose file to report on ---
ReportFile = ""

lFileOpened = OpenFileDialog( lResult, _ ' parent window
"CHOOSE SOURCE FILE", _ ' caption
ReportFile, _ ' filename
PATH_APPLICATION, _ ' start directory
"*.*", _ ' filename filter
".txt", _ ' default extension
nFlags) ' flags

FOR i = 1 TO LEN(ReportFile)
IF ASC(MID$(ReportFile, i, 1)) = 0 THEN
ReportFile = LEFT$(ReportFile, i-1)
END IF
NEXT i

'read data once into array for (multiple) later use

OPEN ReportFile FOR INPUT AS #1
FILESCAN #1, RECORDS TO NumberOfRecords
REDIM ReportDataLine(1 TO NumberOfRecords) AS STRING
LINE INPUT #1, ReportDataLine() TO NumberOfRecords
CLOSE #1

IF PARSECOUNT(ReportDataLine(1), ";") <> %NumberOfGLOBALSUsed THEN

ConsoleMessageBox "This seems to be a file with incorrect format!\n\nReporting is discontinued.",
%OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

ELSE

```



```

'for simplicity reasons the following code assumes the FRA-specific situation of 7 Retail Areas (R1..R7) !!!
'@@@ change here to more generic code

AllRetailAreas = "R1;R2;R3;R4;R5;R6;R7"

ProgressBoxShow %NOCANCEL, 1,%CONSOLE_CENTER, %CONSOLE_CENTER, "Processing "+TRIM$(USING$(###,###,###",
NumberOfRecords))+ " Records.", "Processing file...", %FALSE

FOR CurrentRecord = 1 TO NumberOfRecords

    '--- sum-up according to retail areas and day of week -----
    -----

    lValueFound = %FALSE

    FOR i = 1 TO PARSECOUNT(AllRetailAreas, ";")

        IF PARSE$(AllRetailAreas, ";", i) = PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailAreaActual) THEN

            PAX_ACTUAL_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
PAX_ACTUAL_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_PAX_Actual))

            PAX_ACTUAL_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
PAX_ACTUAL_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_PAX_Actual))

            PAX_ACTUAL_RA(i,8) = PAX_ACTUAL_RA(i,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_PAX_Actual))
            PAX_ACTUAL_RA(8,8) = PAX_ACTUAL_RA(8,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_PAX_Actual))

            Revenue_ACTUAL_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
Revenue_ACTUAL_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_RetailRevenueActual))

            Revenue_ACTUAL_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
Revenue_ACTUAL_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_RetailRevenueActual))

            Revenue_ACTUAL_RA(i,8) = Revenue_ACTUAL_RA(i,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailRevenueActual))
            Revenue_ACTUAL_RA(8,8) = Revenue_ACTUAL_RA(8,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailRevenueActual))

        END IF

        IF PARSE$(AllRetailAreas, ";", i) = PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailAreaSeason) THEN

            'for comparison reason add PAX_Actual and NOT PAX_Season !!!

            PAX_SEASON_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
PAX_SEASON_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_PAX_Actual))

            PAX_SEASON_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
PAX_SEASON_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_PAX_Actual))

            PAX_SEASON_RA(i,8) = PAX_SEASON_RA(i,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_PAX_Actual))
            PAX_SEASON_RA(8,8) = PAX_SEASON_RA(8,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_PAX_Actual))

            Revenue_SEASON_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
Revenue_SEASON_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_RetailRevenueSeason))

            Revenue_SEASON_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
Revenue_SEASON_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_RetailRevenueSeason))

            Revenue_SEASON_RA(i,8) = Revenue_SEASON_RA(i,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailRevenueSeason))
            Revenue_SEASON_RA(8,8) = Revenue_SEASON_RA(8,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailRevenueSeason))

        END IF
    
```

```

THEN
    IF PARSE$(AllRetailAreas, ";", i) = PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_RetailAreaOpti)
    THEN
        'for comparison reason add PAX_Actual and NOT PAX_Opti !!!

        PAX_OPTI_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
PAX_OPTI_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_PAX_Actual))
        PAX_OPTI_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
PAX_OPTI_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_PAX_Actual))

        PAX_OPTI_RA(i,8) = PAX_OPTI_RA(i,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_PAX_Actual))
        PAX_OPTI_RA(8,8) = PAX_OPTI_RA(8,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_PAX_Actual))

        Revenue_OPTI_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
Revenue_OPTI_RA(i,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_RetailRevenueOpti))
        Revenue_OPTI_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) =
Revenue_OPTI_RA(8,VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_DayOfWeek))) + _
VAL(PARSE$(ReportDataLine(CurrentRecord), ";", %FIELD_RetailRevenueOpti))

        Revenue_OPTI_RA(i,8) = Revenue_OPTI_RA(i,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailRevenueOpti))
        Revenue_OPTI_RA(8,8) = Revenue_OPTI_RA(8,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailRevenueOpti))

        INCR RA_Counter(i)

    END IF

    'lValueFound = %TRUE
    'EXIT FOR

NEXT i

'
' IF lValueFound = %FALSE THEN
'     PAX_ACTUAL_RA(9,8) = PAX_ACTUAL_RA(9,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_PAX_Actual))
'     Revenue_ACTUAL_RA(9,8) = Revenue_ACTUAL_RA(9,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailRevenueActual))
'     Revenue_SEASON_RA(9,8) = Revenue_SEASON_RA(9,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailRevenueSeason))
'     Revenue_OPTI_RA(9,8) = Revenue_OPTI_RA(9,8) + VAL(PARSE$(ReportDataLine(CurrentRecord), ";",
%FIELD_RetailRevenueOpti))
'     END IF

    ProgressBarUpdate INT(CurrentRecord/NumberOfRecords*100)

NEXT CurrentRecord

'-----
'--- output -----
'-----

ProgressBarHide

OPEN PATH_APPLICATION+$FILE_DEBUG FOR OUTPUT AS #99

'--- header -----

PRINT #99, "Report generated on " + DATE$ + " at " + TIME$
PRINT #99, "Filename : " + ReportFile
PRINT #99, STRING$(80,"-")
PRINT #99, " "
PRINT #99, " "

'--- detail -----

FOR i = 1 TO 9

    IF i=8 THEN
        PRINT #99, STRING$(80, "-")
    END IF

    PRINT #99, "PAX (ACTUAL) Retail Area      : "; TRIM$(STR$(i)); " = "; USING$("###,###,###",
PAX_ACTUAL_RA(i,8));
    PRINT #99, " i.e. per day --> ";

    FOR j = 1 TO 7
        PRINT #99, USING$("###,###,###", PAX_ACTUAL_RA(i,j)); ";";

```

```

        NEXT j
        PRINT #99, " "
    NEXT i

    PRINT #99, " "
    PRINT #99, " "

    '-----
    -----
    FOR i = 1 TO 9

        IF i=8 THEN
            PRINT #99, STRING$(80, "-")
        END IF

        PRINT #99, "PAX (SEASON) Retail Area   : "; TRIM$(STR$(i)); " = "; USING$("###,###,###",
PAX_SEASON_RA(i,8));
        PRINT #99, " i.e. per day --> ";

        FOR j = 1 TO 7
            PRINT #99, USING$("###,###,###", PAX_SEASON_RA(i,j)); ";";
        NEXT j
        PRINT #99, " "

    NEXT i

    PRINT #99, " "
    PRINT #99, " "

    '-----
    -----
    FOR i = 1 TO 9

        IF i=8 THEN
            PRINT #99, STRING$(80, "-")
        END IF

        PRINT #99, "PAX (OPTI) Retail Area   : "; TRIM$(STR$(i)); " = "; USING$("###,###,###", PAX_OPTI_RA(i,8));
        PRINT #99, " i.e. per day --> ";

        FOR j = 1 TO 7
            PRINT #99, USING$("###,###,###", PAX_OPTI_RA(i,j)); ";";
        NEXT j
        PRINT #99, " "

    NEXT i

    PRINT #99, " "
    PRINT #99, " "

    '-----
    -----
    FOR i = 1 TO 9

        IF i=8 THEN
            PRINT #99, STRING$(80, "-")
        END IF

        PRINT #99, "Revenue (ACTUAL) Retail Area: "; TRIM$(STR$(i)); " = "; USING$("###,###,###,###.##",
Revenue_ACTUAL_RA(i,8));
        PRINT #99, " i.e. per day --> ";

        FOR j = 1 TO 7
            PRINT #99, USING$("###,###,###,###.##", Revenue_ACTUAL_RA(i,j)); ";";
        NEXT j
        PRINT #99, " "

    NEXT i

    PRINT #99, " "
    PRINT #99, " "

    '-----
    -----
    FOR i = 1 TO 9

        IF i=8 THEN
            PRINT #99, STRING$(80, "-")
        END IF

        PRINT #99, "Revenue (SEASON) Retail Area: "; TRIM$(STR$(i)); " = "; USING$("###,###,###,###.##",
Revenue_SEASON_RA(i,8));
        PRINT #99, " i.e. per day --> ";

        FOR j = 1 TO 7
            PRINT #99, USING$("###,###,###,###.##", Revenue_SEASON_RA(i,j)); ";";
        NEXT j
        PRINT #99, " "

```

```

NEXT i

PRINT #99, " "
PRINT #99, " "

'-----
-----

FOR i = 1 TO 9

  IF i=8 THEN
    PRINT #99, STRING$(80, "-")
  END IF

  PRINT #99, "Revenue (OPTI)   Retail Area: "; TRIM$(STR$(i)); " = "; USING$( "###,###,###,###.##",
Revenue_OPTI_RA(i,8));
  IF i >=1 AND i<8 THEN
    PRINT #99, "Flights="; USING$( "#####", RA_Counter(i));
  END IF
  PRINT #99, " i.e. per day --> ";

  FOR j = 1 TO 7
    PRINT #99, USING$( "###,###,###,###.##", Revenue_OPTI_RA(i,j)); ";";
  NEXT j
  PRINT #99, " "

NEXT i

'-----
-----

CLOSE #99

SHELL "notepad.exe " + PATH_APPLICATION+$FILE_DEBUG

END IF

CALL LogEntry(FUNCNAME$, "END")

END SUB 'OPTI_Report()

'-----
-----

SUB OPTI_Avoid_RetailArea_PAX_OverLoad(BYVAL TimeInterval AS INTEGER)

  LOCAL AllRetailAreas AS STRING
  LOCAL RA_MaxPAX AS STRING
  LOCAL RA_CurrentPAX AS LONG
  LOCAL i AS INTEGER

  'for simplicity reasons the following code assumes the FRA-specific situation of 7 Retail Areas (R1..R7) !!!
  '@@ change here to more generic code
  AllRetailAreas = "R1;R2;R3;R4;R5;R6;R7"
  'RA_MaxPAX = "10220;1675;5226;1141;5638;1380;4041" 'implies 1.5 sqm per pax
  'RA_MaxPAX = "10220;1675;5226;1141;5638;1380;4041" 'implies 1.5 sqm per pax
  RA_MaxPAX = "10998;2813;7251;2541;5639;1380;4041" 'implies 1.5 sqm per pax after re-definition of R2 and R3

  FOR i = 1 TO PARSECOUNT(AllRetailAreas, ";")

    RA_CurrentPAX = OPTI_Determine_RetailArea_PAX_Load(i, TimeInterval)
    IF RA_CurrentPAX > VAL(PARSE$(RA_MaxPAX, ";",i)) THEN

      '
      ' ConsoleMessageBox "PAX LOAD is TOO HIGH !!!\n\n" + _
      ' "Time Interval: " + STR$(TimeInterval) + "\n" + _
      ' "Retail Area Index: " + STR$(i) + "\n" + _
      ' "PAX (max) = " + PARSE$(RA_MaxPAX, ";",i) + "\n" + _
      ' "PAX (current) = " + STR$(RA_CurrentPAX) + "\n", _
      ' %OKONLY+%EXCLAMATIONBOX,"WARNING",%IDI_EXCLAMATION,0

      IF TimeInterval < 288 THEN
        'block all gates in this retail area for the next time interval
        CALL OPTI_BlockGatesInRetailAreaInTimeInterval(i, TimeInterval+1)
      END IF

    END IF

  NEXT i

END SUB 'OPTI_Avoid_RetailArea_PAX_OverLoad()

'-----
-----

```

```

FUNCTION OPTI_Determine_RetailArea_PAX_Load(BYVAL RA_Index AS INTEGER, BYVAL TimeInterval AS INTEGER) AS LONG

    LOCAL RelativeGateIndex AS INTEGER
    LOCAL GateIndex AS INTEGER
    LOCAL FlightIndex AS INTEGER
    LOCAL TempTotalPAX AS LONG

    TempTotalPAX = 0

    'for all gates in retail area
    FOR RelativeGateIndex = 2 TO PARSECOUNT(gaRetailArea(RA_Index),";")

        GateIndex = VAL(OPTI_Pheno2Geno_Gate(PARSE$(gaRetailArea(RA_Index),";", RelativeGateIndex)))

        FlightIndex = gGateTime_Matrix(TimeInterval, GateIndex)
        'if there is a flight, then sum up the PAX of that flight
        IF FlightIndex <> %NoFlightValue AND _
            FlightIndex <> %BufferValue AND _
            FlightIndex <> %BlockValue THEN

            TempTotalPAX = TempTotalPAX + VAL(PARSE$(gAllFlightsOnThatDay(FlightIndex), ";", %FIELD_PAX_Actual))

        END IF

    NEXT RelativeGateIndex

    OPTI_Determine_RetailArea_PAX_Load = TempTotalPAX

END FUNCTION 'OPTI_Determine_RetailArea_PAX_Load()

'-----

SUB OPTI_BlockGatesInRetailAreaInTimeInterval(BYVAL RA_Index AS INTEGER, BYVAL TimeIndex AS INTEGER)

    LOCAL RelativeGateIndex AS INTEGER
    LOCAL GateIndex AS INTEGER

    'for all gates in retail area
    FOR RelativeGateIndex = 2 TO PARSECOUNT(gaRetailArea(RA_Index),";")

        GateIndex = VAL(OPTI_Pheno2Geno_Gate(PARSE$(gaRetailArea(RA_Index),";", RelativeGateIndex)))

        'if gate is not yet occupied, block it
        IF gGateTime_Matrix(TimeIndex, GateIndex) = %NoFlightValue THEN
            gGateTime_Matrix(TimeIndex, GateIndex) = %BlockValue
        END IF

    NEXT RelativeGateIndex

END SUB 'OPTI_BlockGatesInRetailAreaInTimeInterval()

'-----

FUNCTION OPTI_Determine_MaxTheoRevenue(BYREF OPTI_GateSet_PerFlight() AS STRING, BYVAL
OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval AS INTEGER) AS DOUBLE

    LOCAL i AS INTEGER
    LOCAL j AS INTEGER
    LOCAL k AS INTEGER
    LOCAL TrialVal AS SINGLE
    LOCAL TrialRA AS STRING

    LOCAL lRA_found AS INTEGER

    LOCAL NumberOfRAs AS INTEGER
    DIM RA_sorted(1 TO 7, 1 TO 2) AS STRING 'REDIM below

    LOCAL TempRevenue AS DOUBLE
    LOCAL TempRACombi AS STRING

    NumberOfRAs = 0

    'sort RA factors
    FOR i = 1 TO UBOUND(gaRF())
        IF gaRF(i,1) <> "" THEN
            INCR NumberOfRAs
        END IF
    NEXT i

    '+++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++
    REDIM RA_sorted(1 TO NumberOfRAs, 1 TO 2) AS STRING
    '+++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++ REDIM +++

    FOR i = 1 TO NumberOfRAs
        TrialVal = VAL(gaRF(i,2))
        TrialRA = gaRF(i,1)

```

```

FOR j = 1 TO NumberOfRAs
  IF TrialVal >= VAL(RA_sorted(j,2)) THEN
    FOR k = NumberOfRAs TO j+1 STEP -1
      RA_sorted(k,2) = RA_sorted(k-1,2)
      RA_sorted(k,1) = RA_sorted(k-1,1)
    NEXT k
    RA_Sorted(j,2) = USING$("##.#", TrialVal)
    RA_Sorted(j,1) = TrialRA
    EXIT FOR
  END IF

NEXT j

NEXT i

TempRevenue = 0

'for each flight
FOR i = 1 TO OPTI_NumberOf_Total_FlightsToBeAllocatedInTimeInterval

  lRA_found = %FALSE

  'test RAs starting with the one with highest revenue
  FOR j = 1 TO NumberOfRAs

    'for each candidate retail area
    FOR k = 1 TO PARSECOUNT(OPTI_GateSet_PerFlight(i,3), ";")

      THEN

        IF GetRetailAreaIndexFromRetailArea(RA_sorted(j,1)) = VAL(PARSE$(OPTI_GateSet_PerFlight(i,3), ";", k))

          'calc revenue for that flight
          TempRevenue = TempRevenue + VAL(PARSE$(OPTI_GateSet_PerFlight(i,4), ";", k))

          'store combi for later use...
          TempRACombi = TempRACombi + PARSE$(OPTI_GateSet_PerFlight(i,3), ";", k) + ";"

          lRA_found = %TRUE

          EXIT FOR

        END IF

      NEXT k

      IF lRA_found = %TRUE THEN
        EXIT FOR
      END IF

    NEXT j

  NEXT i

  IF RIGHT$(TempRACombi,1) = ";" THEN

    TempRACombi = LEFT$(TempRACombi, LEN(TempRACombi)-1) 'remove the last semicolon

  END IF

  gMaxTheoRevenueRACombi = TempRACombi

  OPTI_Determine_MaxTheoRevenue = TempRevenue

END FUNCTION 'OPTI_Determine_MaxTheoRevenue()

'-----

```

## 8.2.2 Include File

```

-----
'--- INCLUDES ---
-----

'#INCLUDE "WINAPI.INC"
'#INCLUDE "PBFORMS.INC"
'#INCLUDE "ComDlg32.inc"

-----
'--- 3RD PARTY (START) ---
-----

'THE CONTENTS OF THIS FILE WERE ORIGINALLY PART OF THE
'WIN32API.INC AND COM32DLG.INC FILES THAT ARE AVAILABLE
'FROM THE POWERBASIC WEB SITE (POWERBASIC.COM). HOWEVER
'THIS CODE HAS BEEN SIGNIFICANTLY MODIFIED TO WORK
'BETTER WITH CONSOLE WINDOWS AND CONSOLE TOOLS.
'
'THIS MODIFIED SOURCE CODE IS PROVIDED AS PART OF CONSOLE
'TOOLS. IT SHOULD NOT BE POSTED IN PUBLIC FORUMS OR
'SHARED WITH NON-LICENSEES.

'THIS SOURCE CODE HAS ALSO BEEN UPDATED TO WORK PROPERLY
'WITH PB/CC VERSION 7.0, WHICH HANDLES ASCIIZ STRINGS
'DIFFERENTLY FROM PREVIOUS VERSIONS OF PB/CC.

%WM_CHILDACTIVATE = &h22
%OFN_EXPLORER     = &h80000
%OFN_ENABLEHOOK  = &h20

TYPE OPENFILENAME
  lStructSize      AS LONG
  hWndOwner        AS LONG
  hInstance        AS LONG
  lpstrFilter       AS ASCIIZ PTR
  lpstrCustomFilter AS ASCIIZ PTR
  nMaxCustFilter    AS LONG
  nFilterIndex      AS LONG
  lpstrFile         AS ASCIIZ PTR
  nMaxFile         AS LONG
  lpstrFileName    AS ASCIIZ PTR
  nMaxFileName     AS LONG
  lpstrInitialDir  AS ASCIIZ PTR
  lpstrTitle       AS ASCIIZ PTR
  Flags            AS LONG
  nFileOffset      AS INTEGER
  nFileExtension   AS INTEGER
  lpstrDefExt      AS ASCIIZ PTR
  lCustData        AS LONG
  lpfnHook         AS DWORD
  lpTemplateName  AS ASCIIZ PTR
END TYPE

DECLARE FUNCTION GETOPENFILENAME LIB "COMDLG32.DLL" ALIAS "GetOpenFileNameA" (lpofn AS OPENFILENAME) AS LONG
DECLARE FUNCTION SETFOREGROUNDWINDOW LIB "USER32.DLL" ALIAS "SetForegroundWindow" (BYVAL hWnd AS LONG) AS LONG

DECLARE FUNCTION OfnHook(BYVAL hdlg AS LONG, BYVAL wParam AS LONG, BYVAL lParam AS LONG) AS LONG

DECLARE FUNCTION OpenFileDialog(BYVAL hWnd AS LONG, _ ' parent window
  BYVAL sCaption AS STRING, _ ' caption
  BYREF sFileNames AS STRING, _ ' filename
  BYVAL sInitialDir AS STRING, _ ' start directory
  BYVAL sFilter AS STRING, _ ' filename filter
  BYVAL sDefExtension AS STRING, _ ' default extension
  BYREF lFlags AS LONG) AS LONG ' flags

-----
'--- 3RD PARTY (END) ---
-----

-----
'--- DECLARATIONS OF SUBS AND FUNCTIONS ---
-----

-----

DECLARE FUNCTION aeroCUBEInit() AS INTEGER
DECLARE FUNCTION TitleBarTime(lNotUsed&) AS LONG
DECLARE SUB SPLASHBOX(sText$)

```

```

DECLARE SUB Test(BYVAL MsgText AS STRING)
DECLARE SUB ShowWaitBox(BYVAL DurationSecs AS INTEGER)
DECLARE SUB ShowDescription()
DECLARE SUB ShowAbout()
DECLARE SUB LogEntry(BYVAL SenderFunction AS STRING, BYVAL LogText AS STRING)

DECLARE SUB Reset_DEBUG_COUNTERS()
DECLARE SUB Show_DEBUG_COUNTERS()
DECLARE SUB Dump_Into_DEBUG_File(BYREF aToBeDumped() AS STRING)

DECLARE FUNCTION ErrorDescription(BYVAL lErrorCode AS LONG, BYVAL lLocation AS LONG) AS STRING

'-----

DECLARE SUB LoadOpsData_Country(BYVAL FileName AS STRING)
DECLARE SUB CalcDelayMinutes(BYVAL FileName AS STRING)
DECLARE FUNCTION DelayMin(BYVAL STD AS STRING, BYVAL ATD AS STRING) AS STRING
DECLARE SUB Flight_DF_Factor(BYVAL FileName AS STRING)
DECLARE FUNCTION GetDutyFreeFactor(BYVAL FlightNo AS STRING, BYREF aDFTable() AS STRING) AS STRING
DECLARE FUNCTION GetDFCountryFactor(BYVAL Country AS STRING, BYREF aDFcountry() AS STRING) AS STRING
DECLARE SUB FillRetailArea(BYVAL FileName AS STRING, BYVAL PurposeIndicator AS INTEGER)
DECLARE FUNCTION GetRetailArea( BYREF RetailAreaDef() AS STRING, _
    BYVAL FltNo AS STRING, _
    BYVAL FltDestCountry AS STRING, _
    BYVAL Gate AS STRING, _
    BYVAL CKI_hall AS STRING, _
    BYREF GatesFound AS LONG, _
    BYREF GatesNotFound AS LONG) AS STRING
DECLARE FUNCTION GetRetailAreaIndexFromRetailArea(BYVAL RetailArea AS STRING) AS INTEGER
DECLARE FUNCTION Is_EU_flight(BYVAL FltDestCountry AS STRING) AS INTEGER

DECLARE SUB FillRetailAreaFactor(BYVAL FileName AS STRING, BYVAL PurposeIndicator AS INTEGER)
DECLARE FUNCTION GetRetailFactor(BYVAL RetailArea AS STRING, BYREF aRFTable() AS STRING) AS STRING
DECLARE SUB SuggestRetailAreaFactor(BYVAL FileName AS STRING)

DECLARE SUB FillPlanningSeason()

DECLARE SUB FillErrorQueue(BYVAL FileName AS STRING)
DECLARE SUB UpdateOAGFile(hWndForm AS DWORD)

DECLARE SUB GenerateWingSpanCode()
DECLARE SUB GenerateAvgGroundTime()
DECLARE SUB FillGroundTime(BYVAL FileName AS STRING)
DECLARE SUB GenerateGatesUsed()
DECLARE SUB ReportStatsPerDay()

DECLARE SUB FillFlightRevenueSpecific(BYVAL FileName AS STRING, BYVAL PurposeIndicator AS INTEGER)

DECLARE SUB ReIndex(BYVAL FileName AS STRING)

DECLARE SUB ReadAllRecsIntoArray(BYVAL FileName AS STRING)
DECLARE SUB WriteRec(BYREF FlightArray() AS STRING, BYVAL LineCounter AS LONG)
DECLARE FUNCTION StringUpdate(BYVAL StringData AS STRING, BYVAL FieldNumber AS INTEGER, BYVAL UpdateText AS STRING) AS STRING

DECLARE SUB GenerateGanttData()
DECLARE FUNCTION GetGateColumn (BYVAL Gate AS STRING, BYREF GateArray() AS STRING ) AS INTEGER
DECLARE FUNCTION GetCategoryABC(BYVAL FieldValue AS STRING, BYVAL ValueDirection AS STRING) AS STRING

DECLARE SUB GenerateWeekDayPAXFile()
DECLARE SUB GenerateWeekDayFIELDFile()
DECLARE FUNCTION SelectField() AS INTEGER

DECLARE SUB DeleteRecords(BYVAL FileName AS STRING, BYVAL FieldNumber AS INTEGER, BYVAL MatchingValue AS STRING)
DECLARE SUB DeleteRecordsTWO(BYVAL FileName AS STRING)
DECLARE SUB FillAssumedATD(BYVAL FileName AS STRING)

DECLARE SUB GenerateTable_AvgOnFieldUnique(BYVAL FileName AS STRING, BYVAL MatchFieldNumber AS INTEGER, BYVAL
ComputeFieldNumber AS INTEGER)

DECLARE SUB FillActualPax(BYVAL FileName AS STRING)
DECLARE SUB ShiftFileIntoHistory(BYVAL FileName AS STRING)
DECLARE SUB Conversion()
DECLARE SUB ConvertCommaToDecimalPoint(BYVAL FileName AS STRING)

'-----

DECLARE SUB OpenBackGroundWindow()
DECLARE SUB GenerateJPGFiles()

DECLARE SUB MyGfxRefresh()
DECLARE SUB ValidMouseLocation (BYREF MyMouseOverX AS LONG, BYREF MyMouseOverY AS LONG)

DECLARE FUNCTION NextDay(BYVAL yyyyymmdd AS STRING) AS STRING
DECLARE FUNCTION DayWeek (BYVAL InDate AS STRING) AS INTEGER

'-----

DECLARE SUB InitAlliances()
DECLARE FUNCTION GetAlliance(BYVAL Airline2ltrCode AS STRING) AS STRING
DECLARE FUNCTION GetTerminalFromGate(BYREF Gate AS STRING) AS STRING
DECLARE FUNCTION GetCKIHallFromGate(BYREF Gate AS STRING) AS STRING
DECLARE FUNCTION GetWingSpanCode(BYVAL AircraftType AS STRING) AS STRING
DECLARE FUNCTION IsRemoteStand(BYVAL Stand AS STRING) AS INTEGER

```





```

%FIELD_ETD = 5 'C
'estimated time of departure ("200603260140")
%FIELD_ATD = 6 'D
'actual time of departure ("20060326015357")
%FIELD_DelayMinutes = 7 'E 'delay
minutes (calculated) btw. ATD and STD ("23", "9999" where over day change)
%FIELD_ACType = 8 'F 'type of
aircraft ("A321")
%FIELD_ACWingSpanCode = 9 'G 'wing span code of aircraft
type
%FIELD_CKIHall = 10 'H 'check-in hall
%FIELD_Terminal = 11 'I 'terminal bldg
%FIELD_Gate_Actual = 12 'J 'gate (as it has been used in the
sample population, real life)
%FIELD_Stand_Actual = 13 'K 'stand (as it has been used
in the sample population, real life)
%FIELD_Gate_Season = 14 'L 'gate (as it has been pre-planned in
seasonal flight planning)
%FIELD_Stand_Season = 15 'M 'stand (as it has been pre-planned in seasonal
flight planning)
%FIELD_Gate_Opti = 16 'N 'gate (as suggested by optimization
run)
%FIELD_Stand_Opti = 17 'O 'stand (as suggested by optimization
run)
%FIELD_PAX_Booked = 18 'P 'number of pax booked on that flight
%FIELD_PAX_Actual = 19 'Q 'number of pax as departed on board
that flight
%FIELD_PAX_Season = 20 'R 'number of pax used for seasonal
flight planning
%FIELD_SeatLF_Actual = 21 'S 'seat load factor (as
observed, real life)
%FIELD_GateSizeActual = 22 'T 'size of boarding gate in square meters (real
life)
%FIELD_GateSizeSeason = 23 'U 'size of boarding gate in square meters
(seasonal planning)
%FIELD_GateSizeOpti = 24 'V 'size of boarding gate in square meters
(optimization run)
%FIELD_StdGroundTime = 25 'W 'standard ground time for that flight
%FIELD_FlightPAXDFfactor = 26 'X 'relative duty free factor specific to a pax on that
flight
%FIELD_DestCountry = 27 'Y 'country of destination ("ES SPAIN")
%FIELD_RetailAreaActual = 28 'Z 'retail area which the gate belongs to
%FIELD_RetailAreaFactorActual = 29 'AA 'relative factor of retail area (determined by
observations)
%FIELD_RetailAreaSeason = 30 'AB 'retail area which the gate belongs to
(but now on basis of seasonal planning gate)
%FIELD_RetailAreaFactorSeason = 31 'AC 'relative factor of retail area (determined by
observations)
%FIELD_RetailAreaOpti = 32 'AD 'retail area which the gate belongs to
(but now on basis of opti run gate)
%FIELD_RetailAreaFactorOpti = 33 'AE 'relative factor of retail area (determined by
observations)
%FIELD_RetailRevenueActual = 34 'AF 'retail revenue for that flight (basis: actual
observations)
%FIELD_RetailRevenueSeason = 35 'AG 'retail revenue for that flight (basis: seasonal
planning)
%FIELD_RetailRevenueOpti = 36 'AH 'retail revenue for that flight (basis: opti run)
%FIELD_TimeIndex = 37 '-- 'STD-Index: generated during opti runs
%FIELD_STAG = 38 '-- 'Scheduled Time At Gate: generated during opti runs
%FIELD_OptiFlightIndex = 39 '-- 'index number of a flight on a day: generated during opti
runs

'-----
'--- ARRAY BOUNDARIES ---
'-----

%AmountOfAirports = 4300
%AmountOfFlights = 12000
%MaxDeparturesPerDay = 1000

'-----
'--- OWN ERROR CODES ---
'-----

%APP_FILE_NOT_FOUND = 160 'in ini-file
%FILE_DOES_NOT_EXIST = 161 'on disk
%WrongNumberOfGlobalVars = 170 'during initialization to check whether .ini-file content and program
code do match

'-----
'--- OTHER INTERN USE ---
'-----

%NumberOfGLOBALsUsed = 39 'number of global variables that are read from ini-file

'test
%IDLABEL = 9999

%NoFlightValue = 8888 ' used in initialization of GateTimeMatrix
%BufferValue = 7777 ' used to indicate a buffer time at the gate
%BlockValue = 6666 ' used to indicate that a gate is not usable (used when PAX load in RA becomes
to high)

%MaxSolutionsToBeTested = 50000 ' max number of valid solutions that may enter the solution stack (if
activated in CombiElements)

```

```
%SolutionStackSize      = 5000 ' number of valid solutions that are stored for later insert trial
%MaxCombiTime           = 60   ' max number of seconds after which a combination run is terminated
```

```
-----
'--- FILES ---
-----
```

```
'--- SYSTEM FILES -----
```

```
$FILE_ACTLOG           = "ACTIVITYLOG.TXT"
$FILE_DEBUG            = "DEBUGFILE.TXT"
$FILE_INI              = "AEROCUBE.INI"
$FILE_PROJECTNOTES    = "PROJECTNOTES.TXT"
```

```
'--- OTHER FILES -----
```

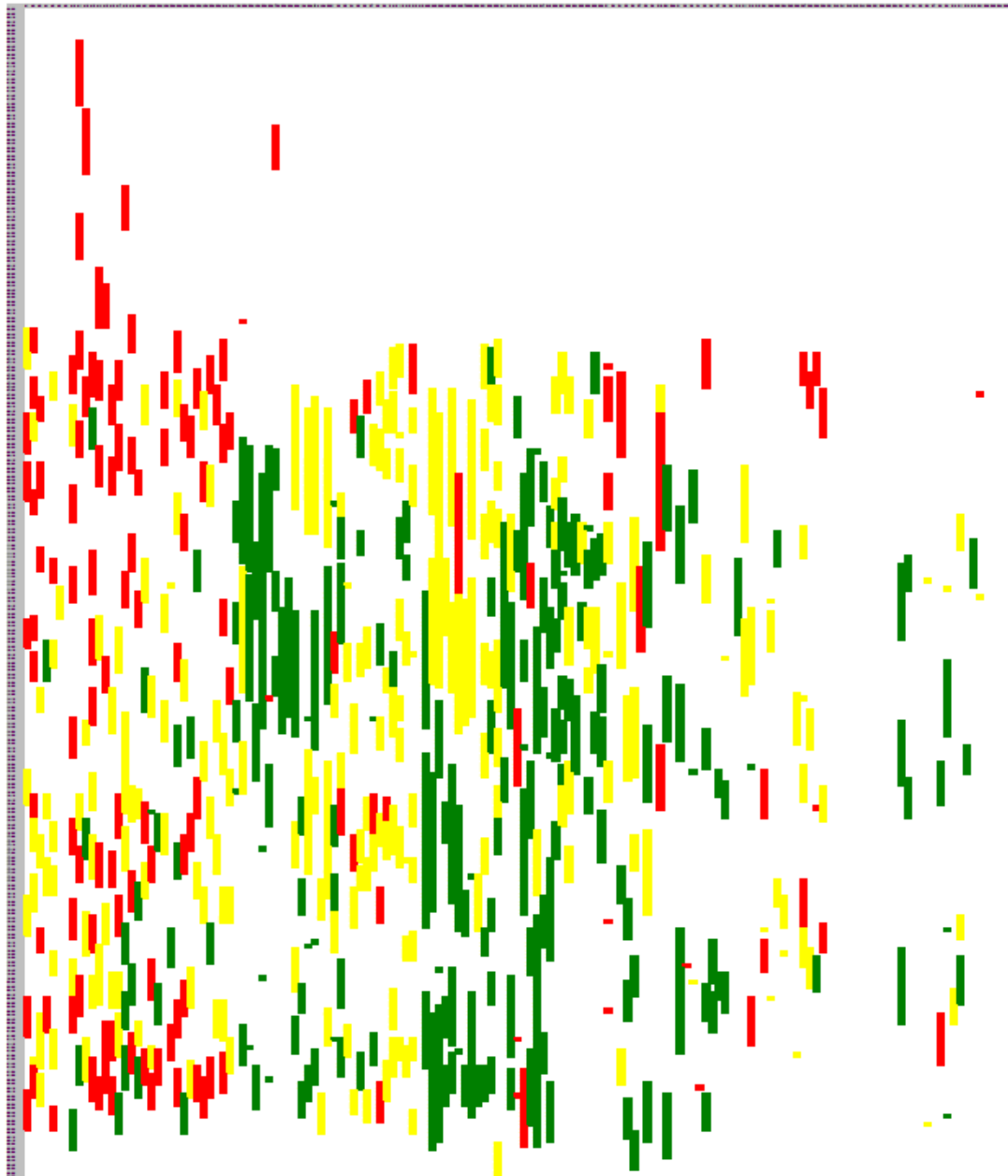
```
$FILE_OUTPUT          = "OUTPUT"
$FILE_OPTIPLAN        = "OPTIPLAN"
$FILE_REVENUES        = "REVENUES"
$FILE_OPTIALLOC       = "ALLOCATION"
```



## 9.2 Sample output of ABC data (formatted)

'Heat map' / 'traffic light' view on retail performance of flights in relation to time and gate (indicator is PAX FLIGHT DF FACTOR)

sample categories: GREEN above 70% YELLOW 35-79% / RED below 35%







Continued with retail information like the retail area, its corresponding retail area factor and the sales generated (for actual, seasonal planning and improved allocation). Further information includes the time a flight is supposed to be at the gate and an internal index.

```
.4;R5;1.4;R7;2.1;113.93;113.93;170.89;183;1425;0334  
T;R3;2.0;R3;2.0;R3;2.0;3169.19;3169.19;3169.19;183;1425;0335  
AB EMIRATES;R5;1.4;R7;2.1;R3;2.0;2574.96;3862.44;3678.51;184;1400;0336  
ITED KINGDOM;R3;2.0;R3;2.0;R3;2.0;2365.84;2365.84;2365.84;185;1430;0337  
;R3;2.0;R3;2.0;R7;2.1;2654.28;2654.28;2786.99;185;1425;0338  
R1;1.0;R3;2.0;R7;2.1;1319.33;2638.66;2770.6;185;1430;0339  
;R2;1.5;R1;1.0;R3;2.0;259.18;172.78;345.57;186;1440;0340  
NGDOM;R5;1.4;R5;1.4;R7;2.1;378.04;378.04;567.06;186;1440;0341  
2;1.5;R2;1.5;R7;2.1;1587.97;1587.97;2223.15;187;1440;0342  
OM;R5;1.4;R5;1.4;R7;2.1;663.12;663.12;994.67;187;1445;0343  
1.0;R3;2.0;R7;2.1;1167.15;2334.31;2451.02;187;1415;0344  
;R1;1.0;R1;1.0;R3;2.0;271.56;271.56;543.13;188;1450;0345  
;R1;1.0;R1;1.0;R3;2.0;172.74;172.74;345.48;189;1455;0346  
;R7;2.1;R7;2.1;302.88;454.32;454.32;190;1450;0347  
R5;1.4;R5;1.4;R7;2.1;219.63;219.63;329.45;190;1500;0348
```





## 9.7 Subset of full report (actual passenger distribution)

|                          |       |                             |            |            |            |            |            |            |            |
|--------------------------|-------|-----------------------------|------------|------------|------------|------------|------------|------------|------------|
| PAX (ACTUAL) Retail Area | : 1 = | 10,763,326 i.e. per day --> | 1,492,339; | 1,475,382; | 1,551,195; | 1,565,449; | 1,638,799; | 1,508,653; | 1,531,509; |
| PAX (ACTUAL) Retail Area | : 2 = | 2,240,802 i.e. per day -->  | 320,504;   | 300,421;   | 316,063;   | 331,461;   | 336,889;   | 305,711;   | 329,753;   |
| PAX (ACTUAL) Retail Area | : 3 = | 8,699,063 i.e. per day -->  | 1,245,584; | 1,185,889; | 1,190,321; | 1,218,446; | 1,313,796; | 1,269,487; | 1,275,540; |
| PAX (ACTUAL) Retail Area | : 4 = | 1,700,401 i.e. per day -->  | 237,360;   | 222,175;   | 236,799;   | 252,677;   | 248,532;   | 252,205;   | 250,653;   |
| PAX (ACTUAL) Retail Area | : 5 = | 2,237,147 i.e. per day -->  | 297,782;   | 304,355;   | 299,992;   | 309,664;   | 365,820;   | 316,759;   | 342,775;   |
| PAX (ACTUAL) Retail Area | : 6 = | 287,480 i.e. per day -->    | 39,515;    | 41,470;    | 41,840;    | 42,535;    | 52,641;    | 31,088;    | 38,391;    |
| PAX (ACTUAL) Retail Area | : 7 = | 1,413,480 i.e. per day -->  | 179,709;   | 175,364;   | 185,757;   | 184,297;   | 240,461;   | 226,392;   | 221,501;   |
| -----                    |       |                             |            |            |            |            |            |            |            |
| PAX (ACTUAL) Retail Area | : 8 = | 27,341,699 i.e. per day --> | 3,812,792; | 3,705,056; | 3,821,967; | 3,904,529; | 4,196,938; | 3,910,295; | 3,990,122; |
| PAX (ACTUAL) Retail Area | : 9 = | 0 i.e. per day -->          | 0;         | 0;         | 0;         | 0;         | 0;         | 0;         | 0;         |

## 9.8 Subset of full report (improved revenue distribution)

|                                 |   |                |                |                |
|---------------------------------|---|----------------|----------------|----------------|
| Revenue (OPTI) Retail Area: 1 = | 2,171,845.76Flights=4819 i.e. per day -->     | 284,965.84;    | 286,006.57;    | 2              |
| Revenue (OPTI) Retail Area: 2 = | 40,823,529.79Flights=36998 i.e. per day -->   | 6,218,126.19;  | 5,360,647.53;  | 5,1            |
| Revenue (OPTI) Retail Area: 3 = | 297,222,179.74Flights=113468 i.e. per day --> | 41,081,235.91; | 40,830,614.26; | 40,            |
| Revenue (OPTI) Retail Area: 4 = | 16,004,994.58Flights=21278 i.e. per day -->   | 2,252,374.18;  | 2,167,094.52;  | 2,1            |
| Revenue (OPTI) Retail Area: 5 = | 3,608.33Flights= 7 i.e. per day -->           | 0.00;          | 0.00;          |                |
| Revenue (OPTI) Retail Area: 6 = | 9,279,035.83Flights=4557 i.e. per day -->     | 1,061,469.96;  | 1,295,714.71;  | 1,1            |
| Revenue (OPTI) Retail Area: 7 = | 124,034,326.68Flights=48303 i.e. per day -->  | 17,198,511.37; | 15,943,374.54; | 16,4           |
| -----                           |   |                |                |                |
| Revenue (OPTI) Retail Area: 8 = | 489,539,520.71 i.e. per day -->               | 68,096,683.45; | 65,883,452.13; | 66,544,429.73; |
| Revenue (OPTI) Retail Area: 9 = | 0.00 i.e. per day -->                         | 0.00;          | 0.00;          | 0.00;          |

## 9.9 Excerpt from summary report of simulation results

Report generated on 01-14-2009 at 23:19:38

Filename : C:\Dokumente und

Einstellungen\Dirk\Desktop\airoCUBEcon\SCENARIOS\All\_New2\SUMMARY\OPTIPLAN\_S\_L.TXT

```
-----
PAX (ACTUAL) Retail Area      : 1 = 10,763,326 ...
PAX (ACTUAL) Retail Area      : 2 =  2,240,802 ...
PAX (ACTUAL) Retail Area      : 3 =  8,699,063 ...
PAX (ACTUAL) Retail Area      : 4 =  1,700,401 ...
PAX (ACTUAL) Retail Area      : 5 =  2,237,147 ...
PAX (ACTUAL) Retail Area      : 6 =   287,480 ...
PAX (ACTUAL) Retail Area      : 7 =  1,413,480 ...
```

...  
...

```
...
PAX (OPTI) Retail Area       : 1 =  1,876,576 ...
PAX (OPTI) Retail Area       : 2 =  4,005,339 ...
PAX (OPTI) Retail Area       : 3 = 12,397,163 ...
PAX (OPTI) Retail Area       : 4 =  2,531,943 ...
PAX (OPTI) Retail Area       : 5 =     5,613 ...
PAX (OPTI) Retail Area       : 6 =   689,358 ...
PAX (OPTI) Retail Area       : 7 =  5,835,707 ...
```

Report generated on 01-16-2009 at 08:17:56

Filename : C:\Dokumente und

Einstellungen\Dirk\Desktop\airoCUBEcon\SCENARIOS\All\_New2\SUMMARY\OPTIPLAN\_S\_M.TXT

```
-----
PAX (ACTUAL) Retail Area      : 1 = 10,763,326 ...
PAX (ACTUAL) Retail Area      : 2 =  2,240,802 ...
PAX (ACTUAL) Retail Area      : 3 =  8,699,063 ...
PAX (ACTUAL) Retail Area      : 4 =  1,700,401 ...
PAX (ACTUAL) Retail Area      : 5 =  2,237,147 ...
PAX (ACTUAL) Retail Area      : 6 =   287,480 ...
PAX (ACTUAL) Retail Area      : 7 =  1,413,480 ...
```

...  
...

```
...
PAX (OPTI) Retail Area       : 1 =  5,086,008 ...
PAX (OPTI) Retail Area       : 2 =  3,261,249 ...
PAX (OPTI) Retail Area       : 3 = 12,470,471 ...
PAX (OPTI) Retail Area       : 4 =           0 ...
PAX (OPTI) Retail Area       : 5 =     5,591 ...
PAX (OPTI) Retail Area       : 6 =   683,454 ...
PAX (OPTI) Retail Area       : 7 =  5,834,926 ...
```

### 9.10 Simulation result: baseline scenario vs. actual traffic (passengers)

| ACTUAL DATA: passengers |           |           |           |           |           |           |           |            |
|-------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|
|                         | Mon       | Tue       | Wed       | Thu       | Fri       | Sat       | Sun       | Σ          |
| R1                      | 1,492,339 | 1,475,382 | 1,551,195 | 1,565,449 | 1,638,799 | 1,508,653 | 1,531,509 | 10,763,326 |
| R2                      | 320,504   | 300,421   | 316,063   | 331,461   | 336,889   | 305,711   | 329,753   | 2,240,802  |
| R3                      | 1,245,584 | 1,185,889 | 1,190,321 | 1,218,446 | 1,313,796 | 1,269,487 | 1,275,540 | 8,699,063  |
| R4                      | 237,360   | 222,175   | 236,799   | 252,677   | 248,532   | 252,205   | 250,653   | 1,700,401  |
| R5                      | 297,782   | 304,355   | 299,992   | 309,664   | 365,820   | 316,759   | 342,775   | 2,237,147  |
| R6                      | 39,515    | 41,470    | 41,840    | 42,535    | 52,641    | 31,088    | 38,391    | 287,480    |
| R7                      | 179,708   | 175,364   | 185,757   | 184,297   | 240,461   | 226,392   | 221,501   | 1,413,480  |
| Σ                       | 3,812,792 | 3,705,056 | 3,821,967 | 3,904,529 | 4,196,938 | 3,910,295 | 3,990,122 | 27,341,699 |

| SCENARIO 'group one, AD-S_F11' (baseline): passengers (sim. result) |           |           |           |           |           |           |           |            |
|---|-----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|
|   | Mon       | Tue       | Wed       | Thu       | Fri       | Sat       | Sun       | Σ          |
| R1  | 213,666   | 228,582   | 255,471   | 243,994   | 258,876   | 213,891   | 224,217   | 1,638,697  |
| R2  | 314,822   | 300,104   | 306,451   | 322,988   | 333,331   | 305,178   | 327,174   | 2,210,048  |
| R3  | 2,158,550 | 2,084,394 | 2,145,048 | 2,201,464 | 2,332,877 | 2,232,878 | 2,217,953 | 15,373,164 |
| R4  | 572,939   | 553,363   | 537,889   | 567,013   | 606,261   | 579,846   | 589,797   | 4,007,108  |
| R5  | 1,338     | 2,564     | 4,784     | 1,076     | 8,250     | 1,059     | 6,557     | 25,628     |
| R6  | 44,911    | 47,603    | 65,384    | 39,927    | 78,152    | 32,041    | 66,760    | 374,778    |
| R7  | 506,566   | 488,446   | 506,940   | 528,067   | 579,191   | 545,402   | 557,664   | 3,712,276  |
| Σ   | 3,812,792 | 3,705,056 | 3,821,967 | 3,904,529 | 4,196,938 | 3,910,295 | 3,990,122 | 27,341,699 |

| SCENARIO 'group one, AD-S_F11' (baseline): passengers (absolute difference) |            |            |            |            |            |            |            |            |         |
|---|------------|------------|------------|------------|------------|------------|------------|------------|---------|
|   | Mon        | Tue        | Wed        | Thu        | Fri        | Sat        | Sun        | Σ          | per day |
| R1  | -1,278,673 | -1,246,800 | -1,295,724 | -1,321,455 | -1,379,923 | -1,294,762 | -1,307,292 | -9,124,629 | -25,068 |
| R2  | -5,682     | -317       | -9,612     | -8,473     | -3,558     | -533       | -2,579     | -30,754    | -84     |
| R3  | 912,966    | 898,505    | 954,727    | 983,018    | 1,019,081  | 963,391    | 942,413    | 6,674,101  | 18,335  |
| R4  | 335,579    | 331,188    | 301,090    | 314,336    | 357,729    | 327,641    | 339,144    | 2,306,707  | 6,337   |
| R5  | -296,444   | -301,791   | -295,208   | -308,588   | -357,570   | -315,700   | -336,218   | -2,211,519 | -6,076  |
| R6  | 5,396      | 6,133      | 23,544     | -2,608     | 25,511     | 953        | 28,369     | 87,298     | 240     |
| R7  | 326,858    | 313,082    | 321,183    | 343,770    | 338,730    | 319,010    | 336,163    | 2,298,796  | 6,315   |
| Σ   | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0       |

| SCENARIO 'group one, AD-S_F11' (baseline): passengers (% difference) |      |     |     |      |     |      |     |        |
|--|------|-----|-----|------|-----|------|-----|--------|
|  | Mon  | Tue | Wed | Thu  | Fri | Sat  | Sun | w.Avg. |
| R1   | -86  | -85 | -84 | -84  | -84 | -86  | -85 | -85    |
| R2   | -2   | -0  | -3  | -3   | -1  | -0   | -1  | -1     |
| R3   | 73   | 76  | 80  | 81   | 78  | 76   | 74  | 77     |
| R4   | 141  | 149 | 127 | 124  | 144 | 130  | 135 | 136    |
| R5   | -100 | -99 | -98 | -100 | -98 | -100 | -98 | -99    |
| R6   | 14   | 15  | 56  | -6   | 48  | 3    | 74  | 30     |
| R7   | 182  | 179 | 173 | 187  | 141 | 141  | 152 | 163    |
| Avg.   |      |     |     |      |     |      |     |        |

w.Avg. = Average based on sum-values of above tables (so: weighted)

### 9.11 Simulation result: baseline scenario vs. actual traffic (sales)

| ACTUAL DATA: retail sales in EUR |            |            |            |            |            |            |            |            |
|----------------------------------|------------|------------|------------|------------|------------|------------|------------|------------|
|                                  | Mon        | Tue        | Wed        | Thu        | Fri        | Sat        | Sun        | Σ          |
| R1                               | 1,860,468  | 1,831,911  | 1,852,479  | 1,830,156  | 1,981,944  | 1,945,155  | 1,989,523  | 13,291,636 |
| R2                               | 451,211    | 411,876    | 434,167    | 455,317    | 461,105    | 417,916    | 475,141    | 3,106,733  |
| R3                               | 6,444,882  | 6,230,744  | 6,217,226  | 6,288,838  | 7,031,121  | 6,748,697  | 6,917,193  | 45,878,702 |
| R4                               | 485,306    | 465,803    | 454,991    | 510,096    | 469,055    | 449,314    | 496,093    | 3,330,657  |
| R5                               | 681,326    | 640,246    | 680,069    | 717,828    | 919,308    | 814,734    | 853,663    | 5,307,175  |
| R6                               | 98,754     | 107,494    | 111,222    | 118,494    | 130,319    | 88,681     | 121,979    | 776,943    |
| R7                               | 929,728    | 877,556    | 901,007    | 817,040    | 1,128,545  | 1,126,778  | 1,122,699  | 6,903,351  |
| Σ                                | 10,951,675 | 10,565,630 | 10,651,160 | 10,737,769 | 12,121,396 | 11,591,275 | 11,976,291 | 78,595,197 |

| SCENARIO 'group one, AD-S_F11' (baseline): retail sales in EUR (sim. result) |            |            |            |            |            |            |            |            |
|--|------------|------------|------------|------------|------------|------------|------------|------------|
|  | Mon        | Tue        | Wed        | Thu        | Fri        | Sat        | Sun        | Σ          |
| R1   | 213,800    | 255,424    | 271,417    | 265,023    | 286,133    | 234,271    | 246,193    | 1,772,261  |
| R2   | 419,857    | 406,573    | 403,402    | 440,561    | 438,432    | 384,889    | 422,578    | 2,916,292  |
| R3   | 8,691,133  | 8,422,647  | 8,463,152  | 8,520,527  | 9,458,274  | 9,284,301  | 9,464,483  | 62,304,516 |
| R4   | 1,255,818  | 1,150,699  | 1,106,885  | 1,159,457  | 1,191,792  | 1,318,465  | 1,283,094  | 8,466,210  |
| R5   | 3,163      | 6,042      | 13,660     | 2,693      | 23,394     | 2,142      | 14,444     | 65,537     |
| R6   | 157,130    | 167,603    | 281,984    | 139,474    | 336,627    | 82,518     | 293,036    | 1,458,370  |
| R7   | 2,092,803  | 1,955,741  | 1,946,961  | 2,074,484  | 2,421,119  | 2,285,114  | 2,319,555  | 15,095,778 |
| Σ  | 12,833,704 | 12,364,728 | 12,487,461 | 12,602,219 | 14,155,770 | 13,591,699 | 14,043,383 | 92,078,966 |

| SCENARIO 'group one, AD-S_F11' (baseline): retail sales in EUR (absolute difference) |            |            |            |            |            |            |            |             |         |
|--|------------|------------|------------|------------|------------|------------|------------|-------------|---------|
|  | Mon        | Tue        | Wed        | Thu        | Fri        | Sat        | Sun        | Σ           | per day |
| R1   | -1,646,667 | -1,576,487 | -1,581,062 | -1,565,133 | -1,695,811 | -1,710,885 | -1,743,331 | -11,519,374 | -31,647 |
| R2   | -31,354    | -5,304     | -30,765    | -14,756    | -22,673    | -33,026    | -52,562    | -190,441    | -523    |
| R3   | 2,246,251  | 2,191,903  | 2,245,926  | 2,231,689  | 2,427,153  | 2,535,604  | 2,547,290  | 16,425,814  | 45,126  |
| R4   | 770,512    | 684,896    | 651,895    | 649,362    | 722,737    | 869,151    | 787,001    | 5,135,553   | 14,109  |
| R5   | -678,163   | -634,204   | -666,409   | -715,135   | -895,914   | -812,593   | -839,219   | -5,241,637  | -14,400 |
| R6   | 58,375     | 60,109     | 170,762    | 20,980     | 206,308    | -6,163     | 171,057    | 681,427     | 1,872   |
| R7   | 1,163,076  | 1,078,185  | 1,045,955  | 1,257,444  | 1,292,574  | 1,158,336  | 1,196,856  | 8,192,427   | 22,507  |
| Σ  | 1,882,030  | 1,799,099  | 1,836,301  | 1,864,450  | 2,034,374  | 2,000,424  | 2,067,092  | 13,483,769  |         |

| SCENARIO 'group one, AD-S_F11' (baseline): retail sales (% difference) |      |     |     |      |     |      |     |        |
|--|------|-----|-----|------|-----|------|-----|--------|
|  | Mon  | Tue | Wed | Thu  | Fri | Sat  | Sun | w.Avg. |
| R1   | -89  | -86 | -85 | -86  | -86 | -88  | -88 | -87    |
| R2   | -7   | -1  | -7  | -3   | -5  | -8   | -11 | -6     |
| R3   | 35   | 35  | 36  | 35   | 35  | 38   | 37  | 36     |
| R4   | 159  | 147 | 143 | 127  | 154 | 193  | 159 | 154    |
| R5   | -100 | -99 | -98 | -100 | -97 | -100 | -98 | -99    |
| R6   | 59   | 56  | 154 | 18   | 158 | -7   | 140 | 88     |
| R7   | 125  | 123 | 116 | 154  | 115 | 103  | 107 | 119    |
| Avg.   | 17   | 17  | 17  | 17   | 17  | 17   | 17  |        |

w.Avg. = Average based on sum-values of above tables (so: weighted)