

## Eduardo Colangelo

»Using Production Logistics Analytics as  
Microservices in Personalized Production«



**Fraunhofer**  
IPA



**Universität Stuttgart**



**Eduardo Daniel Colangelo**

»Using Production Logistics Analytics as Microservices in Personalized Production«

**Herausgeber**

Univ.-Prof. Dr.-Ing. Thomas Bauernhansl<sup>1,2</sup>

Univ.-Prof. Dr.-Ing. Dipl.-Kfm. Alexander Sauer<sup>1,3</sup>

Univ.-Prof. Dr.-Ing. Kai Peter Birke<sup>4</sup>

Univ.-Prof. Dr.-Ing. Marco Huber<sup>1,2</sup>

<sup>1</sup>Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart

<sup>2</sup>Institut für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart

<sup>3</sup>Institut für Energieeffizienz in der Produktion (EEP) der Universität Stuttgart

<sup>4</sup>Institut für Photovoltaik (*ipv*) der Universität Stuttgart

**Kontaktadresse:**

Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA  
Nobelstr. 12  
70569 Stuttgart  
Telefon 0711 970-1101  
info@ipa.fraunhofer.de  
www.ipa.fraunhofer.de

**Bibliographische Information der Deutschen Nationalbibliothek**

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliografische Daten sind im Internet über <http://dnb.de> abrufbar.

Zugl.: Stuttgart, Univ., Diss., 2023

**D 93**

2024

**Druck und Weiterverarbeitung:**

Fraunhofer Verlag, Mediendiensteleistungen, Stuttgart, 2024  
Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.



Dieses Werk steht, soweit nicht gesondert gekennzeichnet, unter folgender Creative-Commons-Lizenz:  
Namensnennung – Nicht kommerziell – Keine Bearbeitungen  
International 4.0 (CC BY-NC-ND 4.0).

# **Using Production Logistics Analytics as Microservices in Personalized Production**

**Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik  
der Universität Stuttgart  
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)  
genehmigte Abhandlung**

**Vorgelegt von**

**Eduardo Daniel Colangelo  
aus Buenos Aires**

Hauptberichter: Univ.-Prof. Dr.-Ing. Thomas Bauernhansl

Mitberichter: Prof. Dr.-Ing. Torsten Kröger

Tag der mündlichen Prüfung: 09.10.2023

Institut für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart

2023



# Foreword of the Author

The application of information technologies has been one of the main innovation drivers in the area of production for several years. However, during my studies and professional career I experienced firsthand the still reigning separation between the manufacturing and digitalization domains. Overcoming this barrier will be essential in order to deal with existing and upcoming challenges. I hope that my work is able to contribute to this objective.

I would like to express my gratitude to my supervising professors, Prof. Dr.-Ing. Thomas Bauernhansl and Prof. Dr.-Ing. Torsten Kröger, for their time and invaluable help. Likewise, I am thankful to the Fraunhofer Institute for Manufacturing Engineering and Automation IPA and the Institute of Industrial Manufacturing and Management IFF of the University of Stuttgart for providing me with the environment and tools necessary.

A doctoral dissertation is, in many ways, a life project spanning several years. As such, its success not only depends on one's individual work but also on a number of people who were willing to offer their support and an occasional helping hand. On a daily basis, I was fortunate to be able to rely on my colleagues: Christian Fries, Leonie Pollmann, Celine Letzger, Theresa-Franziska Hinrichsen, Eftal Okhan, Martina Schaffer, Andreas Kluth, Silke Stump, Susann Kärcher, Christina Berse, Dorothee Böhringer, Dr. phil. Birgit Spaeth, and Dr. phil. Klaus Erlach.

I am also grateful to several colleagues with whom my professional relation was rather short-lived but thankfully not my personal one: Dr.-Ing. Peter Dürr, Dr.-Ing. Marcus Sauer, Dr.-Ing. Erin Sheehan, Dr.-Ing. Andrea Prinz, Prof. Dr.-Ing. Anja-Tatjana Braun, and Dr.-Ing. Thomas Wochinger.

Special thanks go to Dr.-Ing. habil. Hans-Hermann Wiendahl, Dr.-Ing. Silke Hartleif, and Dr.-Ing. Markus Weskamp, who never grew tired of my constant questions.

Lastly, I would like to mention the continuous support of my parents, Jorge Colangelo and Liliana Dirazar, who were with me the whole time in spite of the distance between us.

Stuttgart, November 2023

Eduardo Daniel Colangelo



# Table of Contents

Abbreviations .....	11
List of Figures .....	15
List of Tables.....	19
Kurzfassung.....	21
Short Summary.....	23
1 Introduction and Motivation .....	25
1.1 Current Situation.....	25
1.2 Problem Description .....	26
1.3 Research Question and Approach.....	27
1.3.1 Focus .....	29
1.3.2 Boundaries.....	30
1.4 Scientific Positioning.....	31
1.5 Structure of this Work .....	33
2 Manufacturing Aiming to Enable Personalized Production.....	37
2.1 The Mass Personalization Context.....	37
2.1.1 Definitions .....	37
2.1.2 Comparison with Other Approaches.....	39
2.2 Key Figures of Production Logistics .....	44
2.2.1 Production Logistics .....	45
2.2.2 Key Figures .....	46

---

2.2.3	The Funnel Model.....	49
2.3	Elements and Functions of Current PPC Software.....	51
2.4	Requirements of Mass Personalization on a PPC Software Solution.....	56
2.4.1	Main Aspects of PPC.....	56
2.4.2	Limits of Current Approaches.....	59
2.4.3	The Issue of Data Quality.....	64
2.4.4	Summary.....	66
3	Data Analytics to Assist Production Management Systems.....	71
3.1	Basic Concepts of Data Analytics.....	71
3.1.1	Data Analytics-based Approaches.....	71
3.1.2	Complementing Concepts.....	79
3.1.3	Knowledge Discovery in Databases.....	81
3.2	Data Analytics in Practice.....	86
3.2.1	Challenges.....	86
3.2.2	Domain Knowledge.....	88
3.2.3	Smart Data.....	90
3.3	Assistance of Production Management Systems.....	91
3.3.1	Application of Data Analytics in Production Management.....	91
3.3.2	Example for Throughput Time Prediction.....	93
3.3.3	Summary.....	96
4	Review of Relevant IT Architectures.....	101
4.1	Architectures for Data Analytics.....	101
4.1.1	Data Warehouse.....	101
4.1.2	Extended Requirements.....	105

---

4.2	Software Architectures .....	107
4.2.1	Monolithic Approach .....	108
4.2.2	Modularization.....	109
4.2.3	Distributed Systems.....	112
4.2.4	Service-oriented Computing.....	115
4.2.5	Service-oriented Architecture .....	118
4.2.6	SOA Reference Architecture .....	123
4.2.7	Microservices .....	129
4.2.8	Microservices in the SOA Approach .....	134
4.2.9	Software Architectures in Throughput Time Prediction Example .....	137
4.3	Summary .....	140
5	Proposed Solution: Analytical Microservices .....	145
5.1	The Quadruple Mirroring Idea .....	145
5.2	A Service-oriented Solution .....	148
5.2.1	Implementation Aspects as a Microservices/SOA Hybrid .....	148
5.2.2	IT Efficiency of the Approach.....	158
5.3	Elements of the Solution .....	159
5.4	Comparison with Existing Approaches.....	162
5.5	Summary .....	171
6	Constitution of Analytical Microservices and Their Environment .....	173
6.1	The Modelling Language.....	173
6.2	Structure of the Proposed Solution .....	174
6.2.1	Analytical Microservices.....	174
6.2.2	The Analytical Environment .....	177

---

6.3	Analytical Processes .....	180
6.3.1	Fundamentals of Analytical Processes .....	180
6.3.2	Possible Configurations of Analytical Processes .....	184
6.3.3	Further Considerations and Particular Cases .....	190
6.4	Components and Interfaces .....	192
6.5	Description of the Main Utilization Forms .....	196
6.6	Summary .....	203
7	Application Examples and Critical Evaluation .....	207
7.1	Analysis of Application Scenarios .....	207
7.1.1	Formula for Multi Criteria Analysis .....	208
7.1.2	Setting of the Scenarios .....	211
7.1.3	Evaluation of Scenarios .....	220
7.2	Analysis of an Industrial Use Case .....	223
7.2.1	Description of the Industrial Use Case .....	224
7.2.2	Evaluation of the Industrial Use Case .....	225
7.3	Critical Evaluation .....	231
8	Summary and Outlook .....	235
8.1	Summary .....	235
8.2	Outlook .....	237
9	Bibliography .....	239
10	Appendix .....	283
10.1	Appendix A .....	283
10.2	Appendix B .....	285
10.3	Appendix C .....	287

*German quotes have been translated into English by the author.*



# Abbreviations

AaaS	Analytics as a Service
ABB	Architectural Building Block
AHP	Analytic Hierarchy Process
aM	Analytical Microservice
API	Application Programming Interface
ATO	Assemble-to-Order
B2C	Business-to-Customer
B2B	Business-to-Business
B2U	Business-to-User
BDA	Big Data Analytics
BI	Business Intelligence
BT	Between
CBR	Case-based Reasoning
CIP	Continuous Improvement Process
DA	Data Analytics
DDD	Domain-Driven Design
DSS	Decision Support Systems
DTO	Design-to-Order
DU	Direct Utilization
DW	Data Warehouse

EAIP	Enterprise Application Integration Pattern
EDP	Electronic Data Processing
ESB	Enterprise Service Bus
ETL	Extract, Transformation, and Load
ETO	Engineer-to-Order
FU	Further Utilization
GI	General Investments
GT	Greater Than
IaaS	Infrastructure as a Service
ID	Identifier
IoT	Internet of Things
IS	Information Systems
ISR	Information Systems Research
IT	Information Technology
KDD	Knowledge Discovery in Databases
LT	Less Than
MADM	Multi(ple) Attribute Decision Making
MAUT	Multi Attribute Utility Theory
MCDM	Multi(ple) Criteria Decision Making
MRP	Material Requirements Planning
MTO	Make-to-Order
MTS	Make-to-Stock
OEE	Overall Equipment Effectiveness



---

OLAP	Online Analytical Processing
OR	Operations Research
PaaS	Platform as a Service
PAM	Process of Analytical Microservices
PCS	Process Coordinating Service
PIM	Platform Independent Model
PPC	Production Planning and Control
PPS	Produktionsplanung und -steuerung
PROMETHEE	Preference Ranking Organization Method for Enrichment Evaluations
PTO	Purchase-to-Order
RA	Reference Architecture
RBF	Radial Basis Function
RC	Running Costs
ROI	Return on Investment
SaaS	Software as a Service
SOA	Service-oriented Architecture
SMD	Surface-mounted Device
SVM	Support Vector Machines
SVR	Support Vector Regression
THT	Through-hole Technology
UML	Unified Modelling Language



# List of Figures

Figure 1-1: Relationships between volume and variety in manufacturing.....	25
Figure 1-2: From characteristics to implementation aspects.....	30
Figure 1-3: Systematic of the sciences and positioning of the work.....	32
Figure 1-4: Information Systems Research Framework .....	33
Figure 1-5: Structure of this work.....	35
Figure 2-1: Stockpiling strategies.....	40
Figure 2-2: Characterization of production paradigms .....	42
Figure 2-3: The Funnel Model .....	50
Figure 2-4: Functions of production planning and control.....	52
Figure 2-5: Interaction of material requirements planning functions.....	53
Figure 2-6: Interaction of scheduling and capacity planning functions.....	54
Figure 2-7: Allocation of aspects in order management cycle.....	57
Figure 2-8: Morphology of turbulence germs in production planning and control ...	63
Figure 3-1: Phases for data mining techniques.....	74
Figure 3-2: Overview over machine learning approaches .....	76
Figure 3-3: Overview over concepts of data analytics.....	80
Figure 3-4: Knowledge Discovery in Databases .....	81
Figure 3-5: Simplification of the KDD process.....	84
Figure 3-6: Data analytics example for predicting lead time .....	95
Figure 4-1: Basic architecture of a data warehouse.....	102
Figure 4-2: Reference architecture for SOA solutions .....	123

---

Figure 4-3: First hybrid approach .....	136
Figure 4-4: Second hybrid approach .....	137
Figure 4-5: Architectural designs for throughput time prediction example .....	138
Figure 4-6: Summary of reviewed software architectures .....	144
Figure 5-1: The Quadruple Mirroring Idea .....	146
Figure 5-2: Completeness criteria for implementation aspects .....	154
Figure 5-3: Structure of the proposed solution .....	160
Figure 6-1: Data flows and elements of an analytical microservice .....	175
Figure 6-2: Relation with the SOA Reference Architecture .....	178
Figure 6-3: Trivial analytical process .....	181
Figure 6-4: Detailed execution of an analytical process .....	182
Figure 6-5: Overview of possible configurations of analytical processes .....	185
Figure 6-6: Analytical process with multiple versions of aMs .....	186
Figure 6-7: Analytical process with different aMs .....	187
Figure 6-8: Analytical process with a limited variation of aMs .....	188
Figure 6-9: Analytical process with varying number of steps .....	189
Figure 6-10: Analytical process with parallel execution of aMs .....	190
Figure 6-11: Iterative process for reinforcement learning .....	191
Figure 6-12: Simplified component diagram of the solution .....	192
Figure 6-13: Configuration sequence .....	197
Figure 6-14: Training sequence .....	199
Figure 6-15: Application sequence .....	201
Figure 6-16: Retraining sequence .....	202
Figure 7-1: Relations between complexity dimensions and enterprise types .....	215

---

Figure 7-2: Requirements of scenarios.....	216
Figure 7-3: Comparison of cash flows of alternatives 1 to 3.....	227
Figure 7-4: Comparison of cash flows of alternatives 4 to 6.....	228
Figure 7-5: Comparison of cash flows of alternatives 7 to 9.....	229
Figure 7-6: Comparison of cash flows of alternatives 10 to 12.....	230
Figure 7-7: Utility comparison of analytical microservices and tailored solutions .....	232
Figure 10-1: Detailed component diagram of the solution .....	284
Figure 10-2: Comparison of cash flows of alternatives 1 to 3 (ten models).....	288
Figure 10-3: Comparison of cash flows of alternatives 4 to 6 (ten models).....	289
Figure 10-4: Comparison of cash flows of alternatives 7 to 9 (ten models).....	290
Figure 10-5: Comparison of cash flows of alternatives 10 to 12 (ten models).....	291



# List of Tables

Table 2-1: Logistic objectives for the production reference processes .....	47
Table 2-2: Logistic key figures .....	48
Table 2-3: Requirements on a production management software .....	69
Table 3-1: Features of a possible data analytics-based solution .....	99
Table 4-1: Comparison of software architectures.....	142
Table 5-1: Implementation aspects using a microservices/SOA hybrid approach .....	150
Table 5-2: Aspects for analytical design and knowledge discovery .....	155
Table 5-3: Aspects for analytical modelling and knowledge discovery .....	156
Table 5-4: Aspects for analytical modelling and analytics design .....	157
Table 5-5: Comparison criteria for approaches .....	164
Table 5-6: Comparison with existing approaches.....	165
Table 7-1: Quantification of scenario E1 – F1.....	217
Table 7-2: Quantification of scenario E2 – F2.....	218
Table 7-3: Quantification of scenario E2 – F3.....	219
Table 7-4: Quantification of scenario E3 – F4.....	220
Table 7-5: Utility of the microservices-based solution in the evaluated scenarios.....	221
Table 7-6: Utility comparison of the different approaches.....	222
Table 7-7: Parameters of the industrial use case .....	226
Table 10-1: Factors and weights for the evaluation of application scenarios .....	285





# Kurzfassung

Für den Erhalt ihrer Wettbewerbsfähigkeit wird es für viele Unternehmen in den kommenden Jahren entscheidend sein, personalisierte Produkte in vergleichbaren Mengen und Lieferzeiten anbieten zu können wie Massenartikeln. Dies hat nicht nur einen Einfluss auf die Fertigungstechnologien der Unternehmen, sondern stellt auch eine große Herausforderung für die Gestaltung der organisatorischen Funktionen eines Produktionssystems dar, die sich einer wachsenden Komplexität gegenüber sehen. Um dieser Problemstellung begegnen zu können, müssen die zur Unterstützung solcher Funktionen eingesetzten IT-Werkzeuge zu flexiblen und intelligenten Systemen weiterentwickelt werden. Dies betrifft unter anderem Lösungen für die Produktionsplanung und -steuerung (PPS). Diese Arbeit beschreibt einen Ansatz, solche Systeme durch die Anwendung von Data-Analytics-Komponenten zu gestalten.

Obwohl die verwendeten analytischen Verfahren verschiedene Funktionen bieten, die den Umgang mit Komplexität ermöglichen, ist ihre Nutzung oft problematisch. Dies ist meist auf ihre Implementierung als Software-Monolithen zurückzuführen. Die analysierte Lösung stellt anhand von Microservices hingegen eine flexible Struktur bereit, die sowohl den Anforderungen der Produktionsumgebung als auch den eingesetzten Data-Analytics-Komponenten gerecht wird. Ein Referenzmodell der Lösung mit den Hauptfunktionalitäten wird im Rahmen der Arbeit abgeleitet und veranschaulicht.

Die Arbeit schließt mit einer kritischen Würdigung des entwickelten microservice-basierten Ansatzes durch einen Vergleich mit anderen relevanten Lösungen in verschiedenen Anwendungsszenarien.



# Short Summary

Enterprises that want to remain competitive in the near future need to be able to offer personalized products in the same quantities and delivery times as for mass produced articles. While this certainly presents a challenge to the manufacturing technology, the required organizational functions are of equal (or even greater) importance. To keep up with increasing complexity, the IT tools supporting these tasks – such as production planning and control (PPC) solutions – must evolve into flexible and intelligent systems.

This work describes a novel approach for using data analytics-based components to create such systems. Although these analytical functionalities offer several capabilities for dealing with complexity, applying them successfully is problematic, which is mainly due to their implementation as software monoliths. The solution presented to these problems is to make use of microservices to provide a flexible and evolving structure that matches the requirements of both the production environment and the employed analytical functionalities. A reference model of this solution containing the main functions is derived and illustrated.

The work concludes with a critical evaluation of the microservices-based approach, comparing it to other possible solutions in several application scenarios.



# 1 Introduction and Motivation

## 1.1 Current Situation

Over the years, manufacturing enterprises have developed different types of production strategies to deal with changing markets. This development takes its starting point at around 1850, when the need for complex products in large quantities drove craft production to its limits. Now, 170 years later, the demand for individualization and the strive to remain competitive (Kumar 2007, p. 539; Piller 2008, pp. 136-138) are forcing companies to adopt the paradigm of Personalized Production (see Figure 1-1).

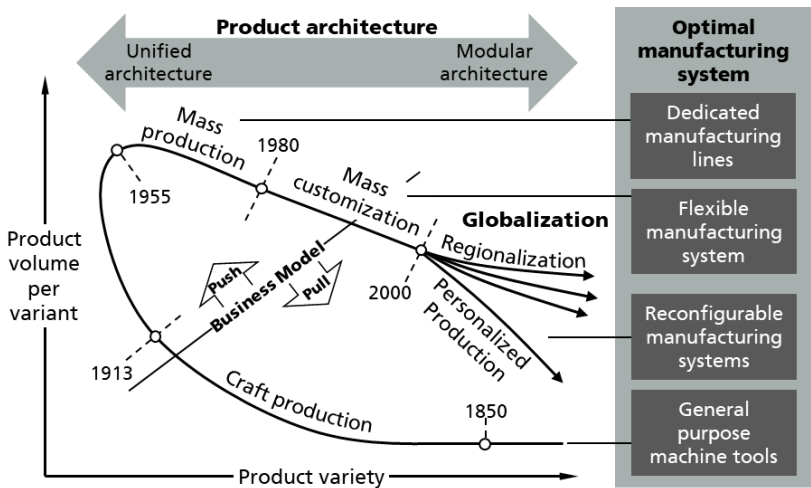


Figure 1-1: Relationships between volume and variety in manufacturing

(Koren 2010, p. 38)

But this phase is not like any other before. Not only does it demand a true personalization of the products (distancing itself from mass customization) (Mourtzis et al. 2014, p. 3) but

it also calls for prices and delivery times comparable to those of mass production (Kumar 2007, p. 537; Hu 2013, p. 7).

Therefore, the changes faced by manufacturing enterprises not only comprise the offering of personalized products and services, but also the creation of manufacturing systems that are able to produce according to the requirements of both customers and manufacturers.

## 1.2 Problem Description

Current initiatives recognize the problems and opportunities created by Personalized Production. However, they mainly concentrate on the development of process technologies – for example, Fischer et al. (2019, pp. 179-183), Keller et al. (2018-2018, pp. 303-309), and Rajamani et al. (2021, pp. 1-19). However, the challenges faced by production management systems, particularly by production planning and control systems, also have to be considered (Wehner et al. 2016, pp. 142-145).

As the challenges presented by Personalized Production are characterized by an increase in complexity, approaches based on the use of data analytics prove adequate (Colangelo et al. 2016a, p. 851; Colangelo et al. 2018, pp. 191-193). Although the potential has been recognized, current initiatives present limitations:

- The use of data analytics (not to be confused with “Big Data”) as an enabler for personalization has been acknowledged, but it is mostly proposed for market-based predictions (demand), relegating logistic problems to a secondary level (Wehner et al. 2016, pp. 156-157).
- Data analytics-based approaches that deal with issues of production logistics are implemented in isolation, lacking an integrated view. Solutions are usually developed to address a specific issue within the big system of interrelated processes and functionalities of production management – for example, as shown by the review performed by Wang et al. (2016, p. 102) – and with a research character (Kusiak 2017, p. 24). The isolation problem also concerns the required integration with the IT landscape (Wierse et al. 2017, p. 374; Woo et al. 2018, pp. 2194-2195).

- Development, implementation, and utilization costs for data analytics solutions (apart from the basic ones) are not affordable for most enterprises. Additional obstacles are the need for specialists and the difficulty to perceive the benefit of such projects. All of this affects the price/performance ratio of the intended solution (PAC 2014, pp. 24-25). Most efforts to address this problem are insufficient (Colangelo et al. 2018, p. 193).

It is therefore necessary to develop an approach that does not only cover the requirements of production logistics using data analytics but also allows for an easy integration with the IT landscape of production management under economic conditions that most enterprises can afford.

The way the data analytics-based approach is implemented as a software solution plays a predominant role. It not only determines the way it can integrate with the IT landscape, but also defines the capabilities of the analytical support provided (e.g. complexity and adaptabilities of the analytical functionalities) and influences the efforts (and costs) related with implementing and utilizing the solution. The flexibility allowed by the architecture of the software solution is therefore a key factor.

In this regard, a new architectural approach has gained importance in the last years: microservices. This further development of the service-oriented architecture is based on the utilization of very small, independent, and autonomous services. This allows for a great modularity, but also for number of advantages, such as the ability to the freely choose technologies and functionalities thanks to the easy exchange of microservices. The resulting solutions are inexpensive and easy to maintain. Furthermore, the approach also enables an easy integration (Newman 2015, pp. 2-8; Dowalil 2018, p. 128).

### 1.3 Research Question and Approach

With a view to the presented problem, the research question to be answered by this work is as follows:

»How can production logistics in the context of Personalized Production be assisted by data analytics-based software solutions in an effective and efficient manner?«

As the research question consists of several elements, it results necessary to derive a set of sub-questions in order to evaluate all pertaining aspects. These sub-questions are:

- »What aspects of production logistics are relevant and how are they affected by Personalized Production?«

Chapter 2 will first define the characteristics of Personalized Production. Subsequently, it will cover the functions of production logistics (and of related software systems) and how they are influenced by the requirements of Personalized Production.

- »What aspects of data analytics are advantageous for Personalized Production?« and »What elements of data analytics should be considered by the software solution?«

Chapter 3 will describe the main characteristics of data analytics. It will not only focus on the aspects relevant for Personalized Production, but also present the elements important for their implementation, including aspects for improving their functionalities (e.g. integration with IT systems).

- »Which aspects of software design can improve the usability of the data analytics-based solutions for Personalized Production?«

The term *usability* is employed here to describe improvements in both the effectiveness and efficiency of the software solution implementing the data analytics-based approaches. Chapter 4 will describe the IT architectures used in software design and their characteristics, including those affecting their adaptability and IT efficiency. Chapter 5 will then propose the utilization of an architectural approach taking into account the requirements of Personalized Production (including the aspects of production logistics) and data analytics. Chapter 6 will provide further details on the approach and chapter 7 will perform an evaluation of the usability.

To answer the research question, this work presents the following approach:

**A microservices-based constitution, provision, and utilization of data analytics.**

Analytical microservices – as partially presented by Colangelo et al. (2018) – should provide a way to deal with issues of production logistics by means of data analytics while



---

allowing for a reduction of several of the associated costs (infrastructure, development, etc.), thus ensuring and facilitating the usability of the solution. For this purpose, the proposed approach will be driven by two main aspects:

- increase of benefit (as perceived by the user with regard to the degree to which the proposed problem is solved)
- reduction of costs

While the first aspect will influence the effectiveness of the solution positively, the second one will have a positive effect on its efficiency. Derived from these two fundamental aspects, the work hypothesis to answer the research question is established as follows:

*»The construction and usage of production logistics analytics as microservices improves the former's benefit-cost ratio under the requirements of Personalized Production«*

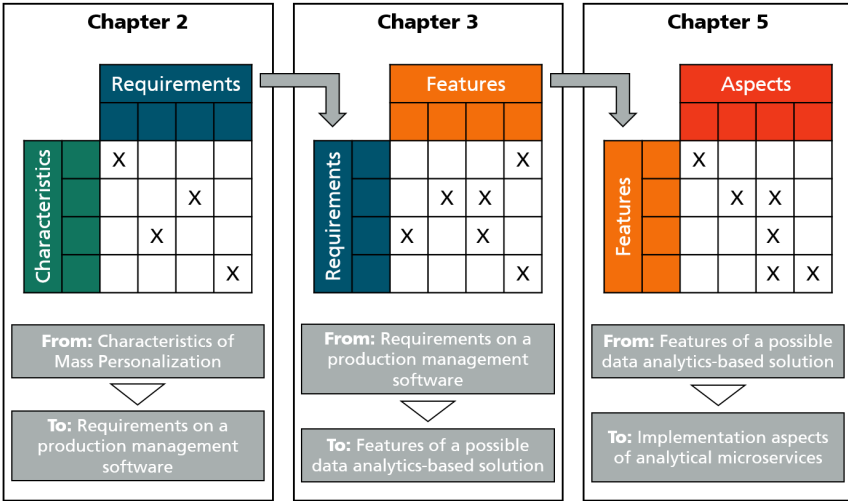
### 1.3.1 Focus

The tasks to be fulfilled within this work are

- to determine the characteristics presented by common issues of production logistics within the context of Personalized Production (but not strictly limited to it) and to derive the corresponding requirements for addressing the said problems;
- to determine the features of data analytics-based solutions to deal with these requirements while taking into account the characteristics and issues of the analytical solutions themselves;
- to determine the aspects of a microservices-based approach to support the analytical solutions, taking into account the determined requirements and features; and
- to present a recommendation for the construction of a microservices-based solution upon the derived aspects. This will consist of the main constructive components and functionalities.

The first three tasks will be addressed in separate chapters, each concluding with an overview – presented as tables – of the elements determined and derived. The order of

the chapters allows for understanding the path from the characterization of the problems to the aspects of the proposed solution. This is illustrated in Figure 1-2.



**Figure 1-2: From characteristics to implementation aspects**

The resulting recommendation should provide a template to be used in the future construction of services-based structures and environments, especially as part of the expansion of *Industrie 4.0* (Industry 4.0) initiatives and products.

Furthermore, this work will focus on validating the established hypothesis (as described in section 1.5).

### 1.3.2 Boundaries

Complementary to the focus of this work, its limits should also be enumerated. These are:

- This work concentrates on data analytics for production logistics (although the approach may be transferable to other areas).

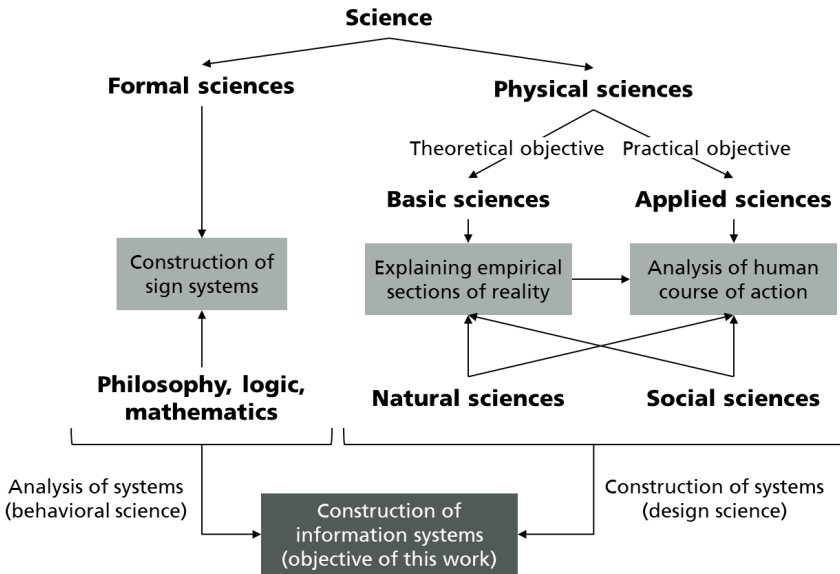
- The focus of this work is on the requirements derived from Personalized Production. However, this does not mean that the approach cannot be used under less demanding circumstances (the application in each new use case must be properly evaluated).
- The usage of data analytics to support production logistics functionalities is presented and analyzed within this work. Nevertheless, it is not the intention to verify this approach, as this is the subject of other current projects and research works (some of which will be cited).
- This work describes only those components and considerations necessary for the construction and utilization of analytical microservices. The following complementary elements are regarded as necessary (and therefore considered) but are not researched in detail: security systems, new methods for data cleaning and transformation, and the design and construction of a marketplace for the distribution of analytical microservices.

#### 1.4 Scientific Positioning

The categorization proposed by Ulrich et al. (1976, p. 305) introduces a first differentiation between physical and formal sciences (see Figure 1-3). While the former are concerned with the description and explanation of perceivable phenomena from the real world, the latter deal with the construction of sign systems (with the corresponding rules that enable their utilization).

Information systems can be defined as sociotechnical systems that support the decision-making process by means of knowledge creation through data processing (WKWI 2011). As such, they are compatible with the objective pursued within this work.

The understanding and construction of information systems is the aim of the German discipline known as *Wirtschaftsinformatik*. This is considered relevant because of its dual approach: it is partially based on formal sciences (for the analysis of systems) and on physical sciences (allowing for the construction of systems) (WKWI 2011).



**Figure 1-3: Systematic of the sciences and positioning of the work**  
(based on Ulrich et al. 1976, p. 305)

The equivalent Anglo-American discipline of *information systems research* (ISR), is more strongly oriented towards behavioral science (a word in this context referring to the application of formal sciences) (Wilde et al. 2007, pp. 280-285). To fulfil the required construction-based vision (taking the role of the physical sciences) it is necessary to concentrate on the branch of ISR known as *design science* (Hevner et al. 2004, p. 76; Wilde et al. 2007, p. 285; Winter et al. 2010, p. 257)

In accordance with the explained duality of information systems, Hevner et al. (2004, pp. 79-80) expresses that while “behavioral science addresses research through the development and justification of theories that explain or predict phenomena related to the identified business need [...] design science addresses research through the building and evaluation of artefacts designed to meet the identified business need”. Behavioral science therefore seeks the truth, while the goal of design science is utility. In his vision, they are inseparable.

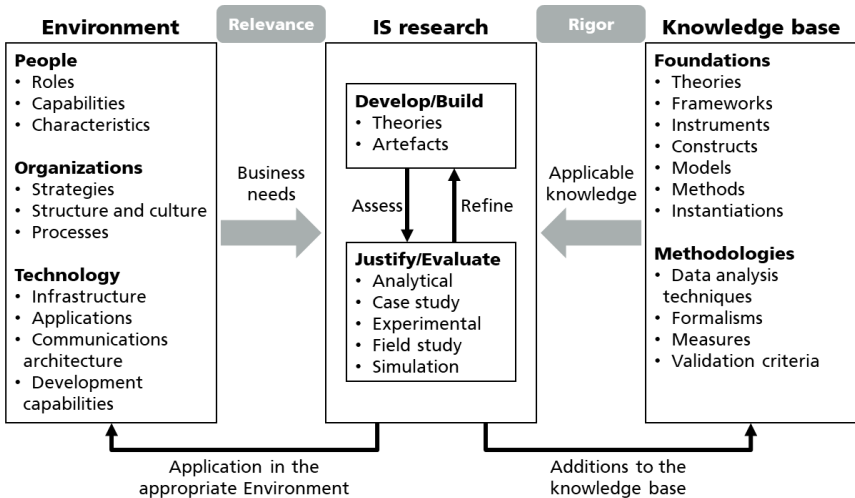


Figure 1-4: Information Systems Research Framework (Hevner et al. 2004, p. 80)

In order to create a research framework, Hevner complements sciences with two principles that guide the ISR: rigor and relevance (see Figure 1-4). While rigor is achieved by appropriately applying existing foundations and methodologies (from the knowledge base, which in turn can also be expanded by research) (Hevner et al. 2004, p. 80), relevance is related to the applicability by a relevant community (Hevner et al. 2004, p. 85).

## 1.5 Structure of this Work

Following the ISR methodology, the structure of this work is based on elements of behavioral and design science. The constructive objective of design science is achieved by applying a methodology proposed by Peffers et al. (2007). The objective of behavioral science is achieved by the application of appropriate methods in the corresponding

sections: mainly in chapters 5 and 6 (modelling of the solution) and chapter 7 (validation of the solution). In this way, the principle of rigor is also fulfilled.

The principle of relevance is considered by analyzing the problem and by researching the state of the art (chapters 1, 2, 3 and 4).

The resulting structure can be viewed in Figure 1-5, with the chapter description on the right side and the corresponding design-science phase (Peffers et al. 2007) on the left side.

The objective of chapters 2 to 4 is to review the state of the art applicable to the problem. They also establish the derivation of requirements on a software-based production management solution to address the characteristics of Mass Personalization (chapter 2) and how these requirements can be met using a data analytics-based solution (chapter 3). Chapter 5 presents the intended microservices-based solution and describes its implementation aspects (as illustrated in Figure 1-2).

As briefly explained in section 1.4, design science requires an *artefact*. IT artefacts are defined as constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems) that allow for the development and examination of IT solutions (Hevner et al. 2004, p. 77; Wilde et al. 2007, p. 281).

From the list of artefacts proposed by Wilde et al. (2007, p. 282) the one chosen in this work is the *reference model*. It fits in with the objective of this work: making a proposal for the constitution and utilization of analytical microservices (using the reference models as a support). The description of the solution (by means of the chosen IT artefact) is given in chapter 6.

For the validation intended in chapter 7, the evaluation method consists of a proposal of scenarios to demonstrate the utility of the proposed artefacts. As described by Hevner et al. (2004, p. 86), the usage of descriptive techniques (of which the utilization of scenarios is one) is meant for cases of innovative artefacts for which other forms of evaluation may

not be feasible. This, therefore, matches the intentions of this work, limited only to the proposal of a solution approach.

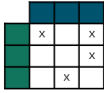
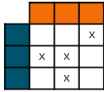
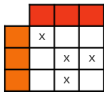
Identify Problem and Motivate	Chapter 1	<b>Introduction and Motivation</b>
Define Objectives of a Solution	Chapter 2	<b>Manufacturing Aiming to Enable Personalized Production</b> Characteristics of Mass Personalization vs. requirements on a production management software 
	Chapter 3	<b>Data Analytics to Assist Production Management Systems</b> Requirements on a production management software vs. features of a possible data-analytics-based solution 
	Chapter 4	<b>Review of Relevant Architectures</b>
Design and Development	Chapter 5	<b>Proposed Solution: Analytical Microservices</b> Features of a possible data-analytics-based solution vs. implementation aspects of analytical microservices 
	Chapter 6	<b>Constitution of Analytical Microservices and Their Environment</b>
Demonstration Evaluation	Chapter 7	<b>Application Examples and Critical Evaluation</b>
	Chapter 8	<b>Summary and Outlook</b>

Figure 1-5: Structure of this work





# 2 Manufacturing Aiming to Enable Personalized Production

This chapter introduces the challenges of Mass Personalization. The focus is not placed on physical implementation (e.g. product configuration solutions, additive manufacturing, etc.) but on how areas of production logistics and the corresponding production management systems are affected. For this purpose, the key elements of production planning and control (PPC) and the functions of the corresponding software solutions are explained as well.

The objective of this chapter is to derive the requirements for software-based production management solutions in order to deal with characteristic situations and problems of Mass Personalization.

## 2.1 The Mass Personalization Context

First, it is necessary to comprehend the actual meaning of Mass Personalization. This section will focus on defining the main terms necessary to understand the intended concept as well as comparing it with other approaches.

### 2.1.1 Definitions

Personalization, being a very promising concept, has different interpretations depending on the area and context it is used in. As this work does not intend a deep analysis of the meaning of personalization, the elements relevant for production systems are extracted.

As seen in Figure 1-1, the current momentum of industrialized society is due to *mass production*. This allows for the production of large volumes of products and the

maximization of productivity, but with a clear disadvantage: the full standardization of the final product (Hu 2013, p. 4). As a way to meet the demand of customized products, approaches based on the utilization of prefabricated modules were developed. This makes it possible to retain most of the benefits of the economy of scale. Such practice is called *mass customization* (Piller 2008, pp. 153-234).

Customization allows, therefore, to address a market segment of a *few*. Personalization, on the other hand, allows to address a market segment of *one* (Kumar 2007, p. 536).

The most successful manufacturing of products that is able to meet all requirements made by customers is based on the *Engineer-to-Order* (ETO) approach. This, however, is normally meant for small quantities (often only one product) and suffers from long lead times and high costs (Kristianto et al. 2013, p. 961). Customers, on the other side, increasingly ask for affordable personalized products, either to cover emerging requirements regarding a more flexible lifestyle, or to better adapt to existing ones (e.g. personalized medicine) (Wehner et al. 2016, pp. 21-27). Enterprises need to be able to personalize in order to become more competitive (Kasanoff 2009).

To understand the first steps towards the desired production of personalized goods in large volumes it is necessary to consider the three views affected: marketing, manufacturing (process), and production management.

Taking them into account, it is possible to differentiate the two main concepts in the context of Personalized Production:

- **Mass Individualization:** This concept encompasses the way products are designed to allow for an individualization according to customer requirements (marketing view). For example, Koren et al. (2015, p. 65) proposes the usage of an open hardware platform in which modules from different sources can be integrated (as opposed to the closed platform of mass customization, where the amount of modules is highly limited).
- **Mass Personalization:** The term *personalization* is considered to have a wider and deeper reach than individualization (Wehner et al. 2016, pp. 32-33). This concept refers not only to the possibility to construct an individualized product but to the actual

creation of products that fulfil the requirements of the customers, even those that are not explicitly communicated (marketing view). Also the technology that enables their production (manufacturing view) and the way that such production systems are to be managed (production management view) in order to meet logistic objectives similar to those of conventional mass production are addressed (Mourtzis et al. 2014, pp. 3-4; Wehner et al. 2016, p. 9). Additionally, the way the term *Personalized Production* is used presents similarities to the characteristics of Mass Personalization (Kumar 2007, p. 536; Hu 2013, p. 7; Mourtzis et al. 2014, p. 3). This work will refer to both Mass Personalization and Personalized Production.

Since much of the literature focuses on personalization in the area of B2C (Business-to-Consumer), this work also intends to cover situations present in B2B (Business-to-Business). The term B2U (Business-to-User) proposed by Wehner et al. (2016, p. 8), which refers to the fulfilment of requirements (independently of B2C and B2B), can also be utilized.

Personalized products not only consist of their tangible part but also (increasingly) of the accompanying services (Piller 2008, p. 137; Wehner et al. 2016, p. 9). Although this work will focus on the creation of physical products, many concepts (such as demand prediction) can be of use in the service area.

## 2.1.2 Comparison with Other Approaches

In order to understand how to fulfil the requirements of Mass Personalization and how similar problems are addressed, it is necessary to review current manufacturing approaches. Following the work of Hoekstra et al. (1992, pp. 6-8), these can be characterized using the different stockpiling strategies, differentiating the areas and activities and whether they are depending on the customer order or not. These, with several additions, can be observed in Figure 2-1.

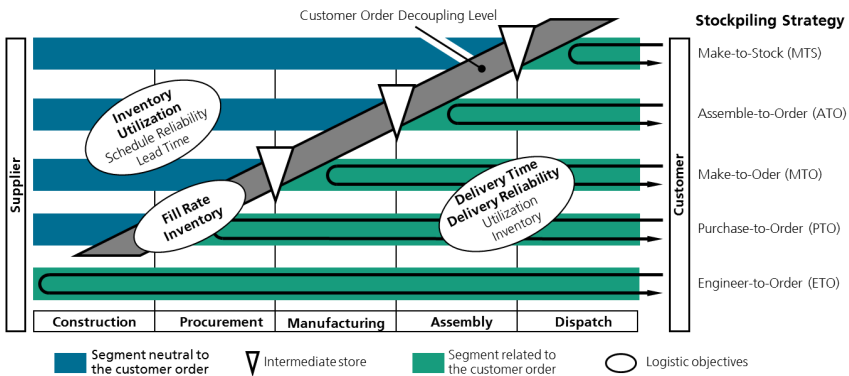


Figure 2-1: Stockpiling strategies (based on Nyhuis et al. 2009, p. 4)

The five strategies are:

- **Make-to-Stock (MTS):** Products are manufactured based on a production program (guided by a demand forecast). It presents the advantage of short delivery times but at the price of elevated stock costs and a high standardization.
- **Assemble-to-Order (ATO):** This strategy prefabricates parts or modules following a production program and performs the final assembly depending on the customer order. It allows for a customization of products (limited personalization) with relatively short delivery times but at the price of elevated stock costs.
- **Make-to-Order (MTO):** In this strategy, manufacturing only starts after receiving a customer order. The raw materials and the semi-finished goods produced externally are purchased in advance, following the disposition parameters (e.g. minimal stock levels) and forecasts. It is usually applied for products that require a higher degree of customization and/or to avoid stock costs. The disadvantage is not only an increase of delivery times but also the reliability of the estimated times and the elevated costs resulting from the necessity of a flexible manufacturing system.
- **Purchase-to-Order (PTO):** Similar to MTO, but different as to some raw materials and semi-finished goods are only purchased when a related customer order exists (usually because of their high value or other problems related to keeping them in

stock). This contributes to an increase in delivery times and deteriorates their estimations. Most systems that claim to produce under a MTO strategy usually use a PTO approach.

- **Engineer-to-Order (ETO):** This strategy allows for the highest degree of personalization, with big parts of the product design influenced by the customer requirements. After finishing the design, production is started, procuring all necessary material and managing the production resources. The disadvantages are clear: customers are facing long lead times before they receive the finalized product, and the price is high (Kristianto et al. 2013, p. 961). Important to consider here is the Design-to-Order (DTO) strategy. The differentiation between ETO and DTO is often not clear, with some authors even using DTO with the same meaning as ETO (Schönsleben 2012, pp. 179-180). This work will consider ETO in general as allowing large changes on existing products (Kristianto et al. 2013, pp. 961-962) and DTO as a subtype of ETO in which the customer is actively involved in the design of a new product (Mandel 2012, p. 30; H.-H. Wiendahl 2020, p. 209).

As mentioned before, Mass Personalization and ETO share the fact that products can be deeply personalized and manufactured in small lot sizes (reaching even lot size one) (Piller 2008, pp. 136-138; Wehner et al. 2016, p. 30). At the same time, there are big differences: the price and delivery time expected by customers are, respectively, much lower and shorter as in ETO (and similar to those in ATO, MTO, and PTO). Furthermore, Mass Personalization works with sales volumes typical of mass production and mostly complex products (Kumar 2007, pp. 534-536; Mourtzis et al. 2014, pp. 3-4).

Manufacturers producing with a Mass Personalization approach will probably combine several approaches (which usually is the case for all strategies described). Hu et al. (2011, p. 729) considers an open product architecture based on three types of modules: *common modules* (shared by all products), *customized modules* (that can be chosen by the customers), and *personalized modules* (specifically created and designed for each customer). Additionally, a distinction can be made between products whose customization and personalization are process-neutral (without changes to the planned

process) and those which are process-specific (requiring new or different operations). It also must be considered that in some cases products and modules, because of their personalization, do not possess a planned process (based on ElMaraghy 2009, pp. 38-39). By extension, a similar distinction can be made in the structure of components and raw materials, having products for which the structure is already known and others whose structure can be configured.

A comparison of the manufacturing paradigms required for the different types of strategies was performed by Mourtzis et al. (2014, p. 4) and can be seen in Figure 2-2. There is no one-to-one correspondence between paradigms and stockpiling strategies, since this is highly dependent on the way an enterprise decides to manage its production system.

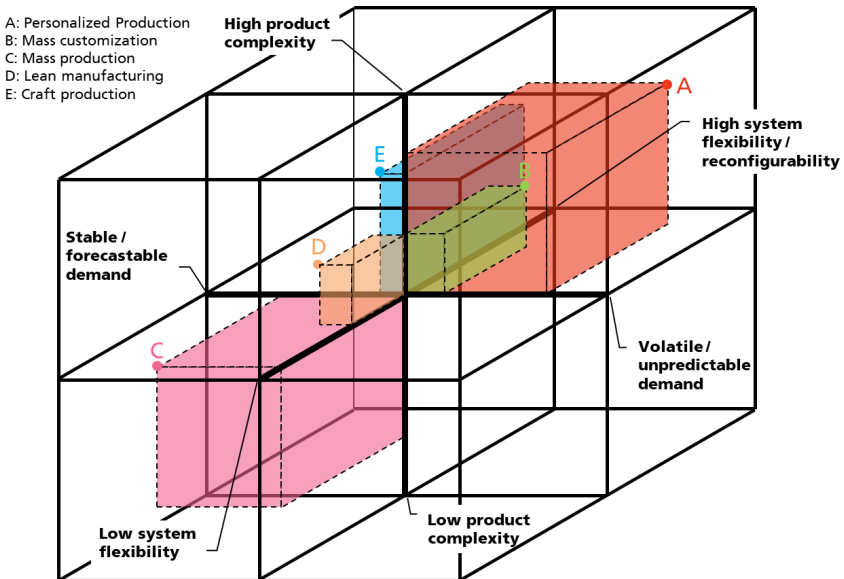


Figure 2-2: Characterization of production paradigms (based on Mourtzis et al. 2014, p. 4)

**Mass production** is heavily influenced by its standardization efforts in order to reduce costs when producing in large volumes. As a result, it possesses a low flexibility and a low complexity of the products offered, what at same time allows for a stable demand (given the low product variety offered, addressing a mass market). **Lean manufacturing** represents “an integrated socio-technical system whose main objective is to eliminate waste by concurrently reducing or minimizing supplier, customer, and internal variability”. Chronologically, it is the next step after mass production, allowing it to deal with the increasing product complexity while improving the flexibility of the manufacturing systems (Shah et al. 2007, p. 791; Mourtzis et al. 2014, pp. 3-4).

**Craft production**, the first manufacturing paradigm introduced by artisans, is still relevant. It can be used, for example, when developing new technologies or serving market niches. It allows manufacturing very complex products with a very flexible manufacturing system (which makes them inefficient and costly). The demand is stable thanks to the very low volume produced (Mourtzis et al. 2014, pp. 3-4).

**Personalized Production** – or, as stated before, Mass Personalization – is located in a particularly challenging octant of the illustration, as it mostly deals with high product complexity, with unknown demand (the term *unpredictable* is avoided, as its implications are too strong), and requires a highly flexible production system. **Mass customization** is a step below Personalized Production in all three dimensions given the lower degree of personalization of the products offered (as explained in section 2.1.1).

The challenge of Mass Personalization can be described using the concept of complexity as developed by Bauernhansl et al. (2014b, pp. 1-4) for production environments. Complexity is defined as having four dimensions: variety (the number of elements), heterogeneity (how different the elements are and the number of such differences), dynamic (how fast the conditions of the environment change), and opacity (the understanding of elements, situations, and their relations; as well as their visibility).

The correspondence with Mass Personalization can then be stated as follows:

- variety with high production volumes
- heterogeneity with elevated personalization of products

- dynamic with high exposure to changes in customer requirements, as well as in manufacturing and sourcing conditions
- opacity with unknown demand, constitution, and production characteristics of the products

Bauernhansl et al. (2014b, pp. 1-4) also proposes that enterprises should deal with complexity by managing it (instead of just avoiding it), as it will be a requirement for them to remain competitive. For a successful management of complexity internal complexity (e.g. products and processes) must correspond with external complexity (originated in the environment). This is based on the **law of requisite variety**, also known as Ashby's law, which states that only "variety can destroy variety" (Ashby 2015, pp. 206-218).

Some of the ways to manage such complexity are part of the objective of this work.

To sum up, the following points are relevant for characterizing Mass Personalization and deriving requirements to a solution:

- the free definition of product features
- small to size one production lots
- a high demand (quantity) similar to mass produced products
- an unknown demand of final products with regard to their configuration and quantity (as well as of the corresponding raw and semi-finished goods)
- similar prices and delivery times as with mass produced products

## 2.2 Key Figures of Production Logistics

This section will deal with defining the concept of production logistics. This, together with the description of the related elements and key figures, will be the basis to understanding the functions of PPC software and the effects of Mass Personalization.



### 2.2.1 Production Logistics

Within this work, the focus will be on the production management view of personalization – which, of course, is strongly dependent on the other two. In this context, *production logistics* is the main concept chosen to encompass the administration and evaluation of the manufacturing system.

In English-speaking countries, the concept of production logistics is mixed up with that of *logistics management*. The latter is defined as “that part of Supply Chain Management that plans, implements, and controls the efficient, effective forward and reverse flow and storage of goods, services, and related information between the point of origin and the point of consumption in order to meet customer requirements” Mentzer et al. (2008, p. 34).

In Germany the definition of the several elements of production and logistics is more detailed. Such is the case with the term *production logistics (Produktionslogistik)*. This reads as follows: “Productions logistics characterizes the phase between procurement logistics and distribution logistics. Part of production logistics are the planning, control, and execution of the transport and storage of raw materials, excipients, supplies, purchased parts, spare parts, semi-finished and finished products; as well as the associated supporting activities within the production system of a company” (Krieger 2018). Given the similarities in the descriptions of the concepts and since several elements to be presented from this point onward originate from the German view, this is the definition that will be used in this work.

The definitions of production logistics also imply, through the control of the material flow, the management of production activities. For example, in order to dispose of a semi-finished product at a specific time, it is necessary to plan all the associated orders, resources, and materials accordingly.

### 2.2.2 Key Figures

As expressed before, delivery times, their reliability, and the price are of outmost importance in Mass Personalization (and in all production strategies) as they are the interface to the customer (Gläßner et al. 1991, pp. 60-64). Production logistics can help to improve these features (assuming that the manufacturing technology is adequate).

This is addressed by Nyhuis et al. (2009, p. 2) who defines the fundamental goal of production logistics as “the pursuance of greater delivery capability and reliability with the lowest possible logistic and production costs. Here, the logistic factor of delivery capability expresses the degree to which it is possible for a company considering the production situation, to commit to the customers preferred delivery date. Delivery reliability on the other hand depicts the extent to which the promised dates for the placed orders can be met”.

From this fundamental goal and the key performance indicators for production firms – delivery capability, delivery reliability, price, logistic process capability, efficiency, and logistic process reliability (Gläßner et al. 1991, pp. 60-64) – the operational objectives of production logistics can be derived (Table 2-1).

The so defined logistic objectives – schedule adherence, throughput time, output rate, and inventory – are contradictory (Nyhuis et al. 2009, p. 9). This conflict is the reason why no optimization is possible, as the proposed solutions to the problems are based on compromises and prioritization of one or several objectives (H.-H. Wiendahl 2012, p. 53). For example, the schedule adherence can be improved at the expense of increasing inventories (and the associated costs).

The “costs” objective can be considered, on a logistic level, as directly dependent on the other objectives.

When categorized according to the main interest, the desired values of the logistic objectives are (H.-P. Wiendahl 1997, p. 136):

- operational goals: high output rate and low inventory
- market goals: high schedule adherence and low throughput time

**Table 2-1: Logistic objectives for the production reference processes (Nyhuis et al. 2009, p. 10)**

		Production reference processes		
		Production and testing	Transport	Storage and supply
Logistic objectives	Schedule adherence	High schedule reliability	High schedule reliability	Low delivery delay
	Throughput time	Short throughput time	Short transport throughput time	Short storage time
	Output rate	High utilization	High utilization	-
	Inventory	Low work in process	Low work in process	Low stock
	Costs	Low costs per unit	Low costs per transport operation	Low storage costs

The “costs” objective belongs to both categories (viewed by the market as price), being in both case the desired value “low”.

To achieve the specified objectives, production logistics make use of three so-called *disposition objects* (H.-H. Wiendahl 2012, p. 65). These are:

- articles (products, materials)
- resources (machines, workers, etc.)
- orders

Orders play a central role, as they contain and summarize all information necessary to control the production: the materials involved (with the corresponding quantities), the resources required, the processes to be followed, the corresponding scheduling data, and all additional parameters. It is worth mentioning that, in this context, the term *orders* refers to production orders and their subtypes (manufacturing, assembly). When referring to other order types (such as sales or purchase), it will be explicitly mentioned.

Order management is the name given to the “set of tasks and areas necessary to plan and optimize the order flow in the production” (Westkämper et al. 2006, p. 179). It is the

central authority responsible for the availability of the required goods, being its main task the assignment of articles, processes, and resources to orders; with regards of time, quantity, and place (H.-H. Wiendahl 2012, p. 63).

Additionally to the disposition objects, H.-H. Wiendahl (2012, p. 65) proposes the division of processes (initially) in three phases based on the three main processes of the Supply Chain Operations Reference (SCOR) model: source, make, and deliver (SCOR 12.0). To simplify, he proposed a joint consideration of make and source (an “external” view of production) under “disposition”.

Based on these two dimensions (disposition objects and process stages) and the logistic objectives, H.-H. Wiendahl (2012, p. 108) proposes a series of key figures to measure the performance of an order management system (and the associated production logistics). These can be found in Table 2-2.

**Table 2-2: Logistic key figures (H.-H. Wiendahl 2012, p. 108)**

		Disposition object		
		Order	Resource	Article
Process phase	Deliver	Absolute values <ul style="list-style-type: none"> <li>• Delivery time</li> <li>• Delivery lot size</li> <li>• Delivery deviation</li> </ul> Relative values <ul style="list-style-type: none"> <li>• Delivery dependability</li> </ul>	Absolute values <ul style="list-style-type: none"> <li>• Sales</li> </ul> Relative values <ul style="list-style-type: none"> <li>• Backlog</li> <li>• Backlog coverage</li> </ul>	Absolute values <ul style="list-style-type: none"> <li>• Missing quantity</li> </ul> Relative values <ul style="list-style-type: none"> <li>• Delivery delay</li> <li>• Fill rate</li> </ul>
	Disposition	Absolute values <ul style="list-style-type: none"> <li>• Lead time</li> <li>• Purchase / manufacturing lot size</li> <li>• Schedule / quantity deviation</li> </ul> Relative values <ul style="list-style-type: none"> <li>• Flow rate</li> <li>• Schedule / quantity reliability</li> </ul>	Absolute values <ul style="list-style-type: none"> <li>• Output</li> <li>• Work in process</li> </ul> Relative values <ul style="list-style-type: none"> <li>• Delivery dependability</li> <li>• Work-in-process coverage</li> </ul>	Absolute values <ul style="list-style-type: none"> <li>• Time a product spends in store</li> <li>• Inventory</li> </ul> Relative values <ul style="list-style-type: none"> <li>• Inventory turnover</li> <li>• Inventory coverage</li> </ul>

In order to influence these key figures, H.-H. Wiendahl (2012, p. 108) mentions four use cases (the first three being directly relevant for the regular operation of a logistic system). These are:

- 
- Reference variables: the specified values to be achieved. They manifest as target (for long periods) and set (for short periods) values, as well as the corresponding process and planning parameters.
  - Correcting variables: the variables that can be directly influenced through operational decisions.
  - Regulating variables: the variables that are monitored and compared to the reference variables during the ongoing operation and that are influenced through the correcting variables. It is necessary to differentiate between actual (related to present) and planned (related to the future) values.
  - Improvement variables: appear during transformation projects and manifest as target and actual values.

To sum up, the following points are relevant for characterizing Mass Personalization and deriving requirements to a solution:

- The logistic objectives and the related key figures are the main way to control the performance of manufacturing systems under the different strategies (including Mass Personalization).
- Productions logistics, responsible for fulfilling the logistic objectives, require functions for continuous management and optimization in order to do so.

### 2.2.3 The Funnel Model

The **Funnel Model**, developed by Leibniz Universität Hannover, allows for the modeling of logistic objectives and the description of production processes (Lödding 2008, p. 50; Nyhuis et al. 2009, p. 17).

The following cycle elements of an operation can be identified (Nyhuis et al. 2009, pp. 17-23):

- **Work content:** a key parameter for the Funnel Model. It is a measure of the planned time for an operation on a work station.

- Throughput time:** in the case of one operation (as it can be seen in Figure 2-3), it is the time span an order requires from the completion of the previous operation – or from the order’s point of input (at the start of an operation) – until the end of the observed operation’s process. In the case of an order, it is the time span between its point of input and the end of its last process. *Throughput time* and *lead time* are often used in the same sense, although the former refers to the internal vision of the manufacturer, while the latter is the customer view (time from purchase to delivery).
- Inter-operation time:** the time between the end of the previous operation and the start of setup of the observed operation’s process. It can be calculated through the addition of the post-process waiting time, the transport time, and the pre-process waiting time. This time is assigned to the observed operation.

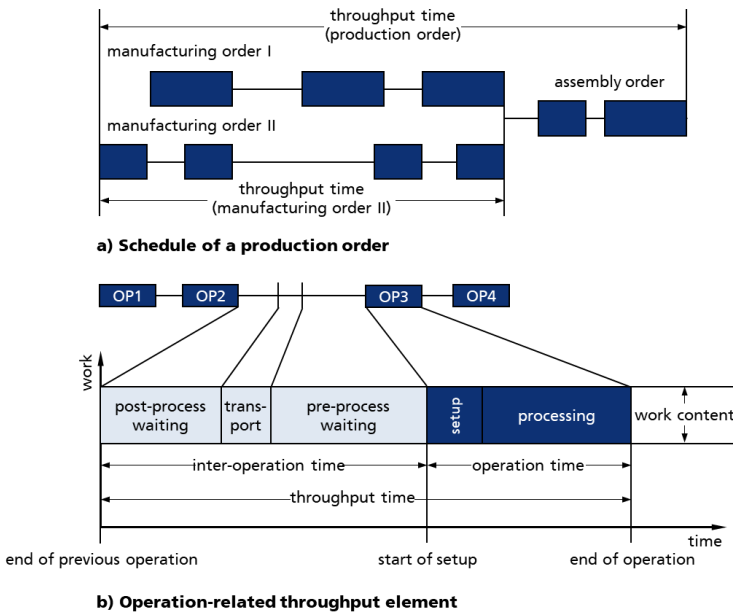


Figure 2-3: The Funnel Model (Nyhuis et al. 2009, p. 22)

- **Operation time:** the time between the start of setup and the end of the observed operation's process. It can be calculated through the addition of the setup time and the processing time. It can be alternatively calculated by dividing the work content by the capacity of the work station (H.-P. Wiendahl 1997, p. 19).

These concepts can also be considered at the superordinate levels to the operation (order or set of orders to produce an article). The values can be calculated based on the data available at the level, or derived from the respective elements of the sub-object (orders, operations).

Often, the values of several elements are not known – for example, setup times are rarely recorded. Due to the additive nature of the elements, it is possible to distinguish three levels (from the lowest to the highest detail):

1. Throughput time
2. Inter-operation time and operation time
3. Post-process waiting time, transport time, pre-process waiting time, setup time, and processing time

One of the main uses of the Funnel Model is the recognition and analysis of deviations (between actual, set, and planned values).

To sum up, the following points are relevant for characterizing Mass Personalization and deriving requirements to a solution:

- the existence of a structure for describing production processes that can be used for management and optimization functions of production logistics
- the possibility in this structure to work with different levels of detail

## 2.3 Elements and Functions of Current PPC Software

After explaining the meaning of production logistics and the related concepts, it is necessary to understand how the corresponding functions are implemented using software solutions.

From a behavioral point of view, PPC can be defined as “a cross-sectional function within the enterprise that deals with operational control of material and information flows. Its task is to allocate all required resources to sales orders in terms of quantity and time” (Schotten 1998, p. 262). Implemented as a software tool, PPC is “the EDP-supported organizational planning, control, and monitoring of production processes from offer processing up to dispatch” (Westkämper et al. 2006, p. 180).

PPC software is responsible for implementing functionalities to perform the tasks of order management, as both share the same objectives (H.-H. Wiendahl 2002, p. 33; Westkämper et al. 2006, p. 180; H.-P. Wiendahl et al. 2020, pp. 246-247).

Area	Main Function	Function	Time horizon
Production Planning	Program Planning	Forecast calculation Management of customer order Rough-cut planning Scheduling of preparation time Determination of delivery date	Long-term
	Material Requirements Planning	Determination of gross requirements Supplier selection Inventory allocation Procurement calculation Determination of net requirements Stock management	
	Capacity Requirements Planning	Disposition Disposition Production order generation Purchase order generation	Middle-term
		Load planning Capacity planning Sequence planning	
Production Control	Order Initiation	Production order release Receipt creation Production order allocation Purchase order release Purchase order processing Purchase order monitoring	Short-term
	Order Monitoring	Production order progress monitoring Quantity and schedule monitoring Quality control Goods receipt monitoring Quantity and schedule monitoring Quality control	

**Figure 2-4: Functions of production planning and control**  
(Hackstein 1989, pp. 3-17; H.-H. Wiendahl 2002, p. 36, 2020, p. 234)

In Figure 2-4 the areas and functions of PPC are shown. These are, based on H.-P. Wiendahl (1997, pp. 12-14) and H.-H. Wiendahl (2020, p. 234):



- **Production planning:** responsible for planning ahead the manufacturing process for a certain period of time in the future.

This area comprises the following functions:

- **Program planning:** This long-term function determines (usually weekly or monthly) the primary requirements based on sales forecasts and existing customer orders, considering the existing capacity. The result consists of a *list of saleable products* by type and quantity for a planning horizon of months to years (depending on the branch).
- **Material requirements planning:** The first middle-term function determines (usually on a daily basis) the requirements for bought-in and in-house manufactured parts in terms of type, quantity, and date, based on the bill of materials. Requirements for the same components on a (configurable) certain period are organized together into lots.

The group of sub-functions responsible for carrying out the planning activity, usually abbreviated MRP, is described in Figure 2-5. The planning process is known as disposition.

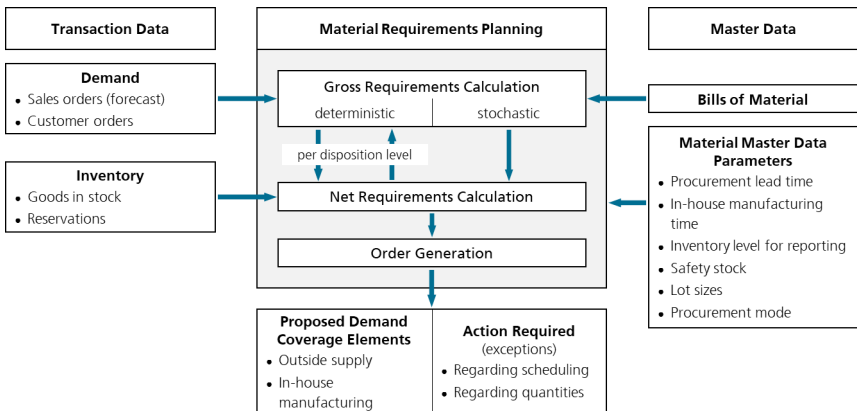


Figure 2-5: Interaction of material requirements planning functions

(H.-P. Wiendahl et al. 2020, p. 295)

Inventories of raw materials, semi-finished parts, and finished products – as well as their foreseeable development over the course of time (reservations) – are taken into account. Furthermore, a set of material-related parameters (e.g. in-house manufacturing time, lot size, etc.) are considered.

The result is the generation of *production and purchase orders* for a planning horizon of days to months (depending on the branch).

- **Capacity requirements planning:** The second middle-term function determines (usually on a daily basis) for each production order the start date based on the planned due date, the operation sequence in the corresponding routing, and other influencing factors (material availability, priorities, etc.). This is referred to as load planning. In subsequent capacity planning, the resulting loading of the resources (machines and workers involved) is checked. The capacity synchronization takes care of load balancing by means of postponements of production orders and capacity alignment. The final step is sequence planning, which fixes the executing sequence of the production orders (Lödding 2008, p. 90). A detailed picture of this planning function is shown in Figure 2-6.

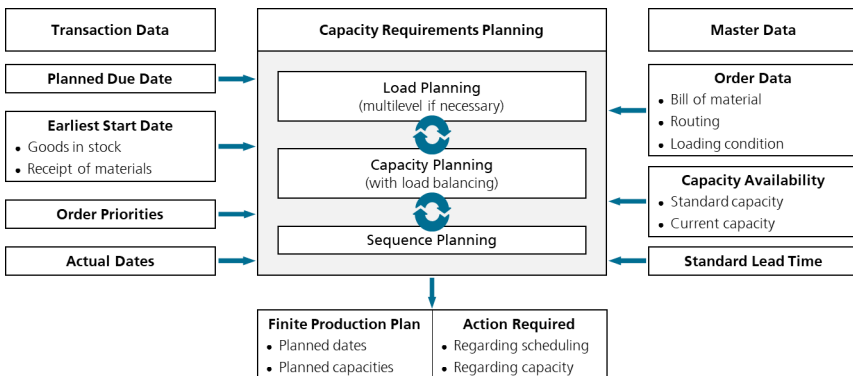


Figure 2-6: Interaction of scheduling and capacity planning functions

(H.-P. Wiendahl et al. 2020, p. 325)

The resulting *finite production plan* (with precise planning of production orders in terms of time and resources) refers to a planning horizon of days to weeks (depending on the branch).

- **Production control:** responsible for the realization of the plan while dealing with unavoidable changes regarding production order quantity and date as well as disturbances due to staff shortages and machine breakdowns.

This area comprises the following functions:

- **Order initiation:** This short-term function confirms and makes adjustments to the assignment of production orders to individual machines and work stations based on current conditions (e.g. short-term capacity and material availability). This results in the *loading plan and schedule*, which refers to a planning horizon of days to weeks (depending on the branch).
- **Order monitoring:** This short-term function undertakes continuous supervision. The feedback obtained not only allows for adjusting the plans (H.-H. Wiendahl 2002, p. 84) and creating key figures, it is also necessary to adjust the master data utilized for planning.

Regarding **data management**, two types of data are to be considered (as shown in Figure 2-5 and Figure 2-6) (Benz et al. 2011, p. 59):

- **Master data** remain unchanged over a long period of time. They contain information repeatedly required in an identical manner. Examples are material data, bills of material, routings, theoretical capacity, etc.
- **Transaction data** originate from a process (business transaction). Orders, inventory, current capacity, and feedback data are examples of transaction data. They often utilize master data as a basis (e.g. production orders are created based on the operation sequence contained in a routing).

This functionalities can be found in one IT solution or (more usually) distributed among several (ERP, MES, APS) working together (ANSI/ISA 95; VDI 5600 Blatt 1).

To sum up, the following points are relevant for characterizing Mass Personalization and deriving requirements to a solution:

- the existence of planning and control functions to fulfil the management and optimization requirements of production logistics
- the utilization of different levels of detail and time horizons in each function
- the employment of master and transaction data as a basis for performing the PPC functions

## 2.4 Requirements of Mass Personalization on a PPC Software Solution

Kletti (2007, p. 22) suggests that every enterprise is a permanently disturbed system: “every turn of the machine and every call of the customer changes the conditions”. Disturbances are responsible for variations in the planned time flows and stocks, affecting the logistic objectives (Westkämper et al. 2000, p. 844) and causing the need to permanently update the planning scenario.

Additionally to internal disturbances (machine breakdown, staff shortage, product quality, etc.), PPC systems also face external factors (e.g. unreliable suppliers) over which they do not have any control.

This section will explain the challenges faced by PPC software and how their difficulty is further increased by Mass Personalization. Towards the end of the chapter, the emerging requirements on a PPC software solution are described.

### 2.4.1 Main Aspects of PPC

To understand the requirements of PPC software solutions and how they are addressed, a categorization of the functions based on their main content viewed from an analytical standpoint is proposed. Building upon the approach of H.-H. Wiendahl (2002), two constitutive aspects of PPC functions are considered:

- forecasting aspect

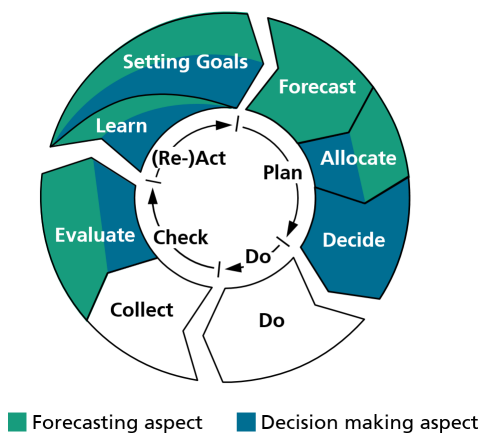
- decision-making aspect

Although both concepts are deeply intertwined with each other (decision-making always involve some sort of forecasting), the separation helps to understand the form that the corresponding functionalities should take.

Both aspects are relevant in planning: forecasts are made to foresee future states (demand, material availability, real capacity, etc.) and subsequent decisions (products to be produced, allocation of resources, etc.) are embodied as resulting plans.

The consideration of both aspects within the control area is of special interest to this work. Control functions can be viewed as working on a control loop: data are gathered from the work environment and decisions are made (e.g. Is it necessary to change the current plan? Are the variables satisfying? etc.) based on an analysis to understand the current situation and the prediction of future states (forecasting). These decisions may eventually trigger new planning activities.

H.-H. Wiendahl proposed in Westkämper et al. (2000, pp. 847-851) to apply Deming's PDCA (plan, do, check, act) approach to constitute the order management cycle (Figure 2-7).



**Figure 2-7: Allocation of aspects in order management cycle (based on H.-H. Wiendahl in Westkämper et al. 2000, p. 849)**

As shown in Figure 2-7, *forecast* and *decide* are not the only phases present within the cycle. Some considerations are therefore necessary:

- *Allocate* is responsible for assigning resources based on foreseeable requirements and usage (thus addressing both aspects).
- The *learn* phase is similar as both, the forecasting and the decision aspect, require learning.
- *Evaluate* comprehends decision-making based on the analysis of the current situation and predictions of future states (thus addressing both aspects).
- The *setting of goals* is performed as a result of the decisions made and serves as an input for the creation of forecasts.

Forecasting and decision-making both require the *collection* of data (feedback) in order to act based on reality and as input for the learning phase. However, they do not act operatively during this phase and for this reason they are left blank. It is similar with the *do* phase, which represents execution based on the results of forecasting and decision-making aspects.

Although the cycle possesses well-defined steps such rigid structure could also be problematic. Analysis, forecasts, and decisions based on the current situation and available data are continuously required. Furthermore, learning should assist as many functions as possible and can be seen as a permanently running process. Such vision responds to a modern notion of software solutions (with objects and parallel processes instead of structured procedures) and is highly influenced by the requirements of Mass Personalization regarding flexibility.

To sum up, it is relevant for characterizing Mass Personalization and deriving requirements to a solution to consider the existence of forecasting and decision-making aspects in PPC software, as well as the requirements of PPC functions emerging from said aspects and how these requirements can be addressed.

### 2.4.2 Limits of Current Approaches

The functionalities responsible for implementing the two aspects in PPC systems are based on algorithms and models. Originally deterministic algorithms – in which the output depends only on the input (Alpaydin 2016, p. 32) – were extended to consider stochastic elements by adding statistics-based components (e.g. mean values, regression, exponential smoothing, etc.) (Günther et al. 2016; Tempelmeier 2016; Colangelo et al. 2018, p. 193), thus turning into randomized algorithms (Motwani et al. 2007, pp. x-xi). These are, however, still subject to limitations.

Extending the ideas of H.-H. Wiendahl (2002, p. 100), the following application of algorithms and models can be considered in order management activities:

- forecast
- optimization
- supporting analysis

While the first one clearly belongs to the forecasting aspect, optimization is, at the bottom line, a decision-making process. Supporting analyses are necessary for both, the forecasting and the decision-making aspect.

In PPC functionalities dealing with optimization problems, the usage of heuristic algorithms is common – Zhang (2013) provides an example for scheduling. Originating in operations research (OR), heuristic methods use experience or judgement to solve problems (although they do not guarantee an optimum) (Aickelin et al. 2011, p. 251).

Typical applications for the mentioned algorithms and models are:

- **Throughput time determination** can be affected by a many factors, both random and caused by PPC decisions (e.g. false prioritization of orders). This makes it an aleatory variable (Tempelmeier 2016, p. 317). As seen in the funnel model, throughput time can be determined alone or based on its constitutive elements (each one with its own sources of uncertainty). One of the reasons for the importance of this value is its application in the estimation of delivery time.

- **Capacity determination** is one of the more difficult estimations. Capacities are not only affected by a wide range of factors; the further away planning is from the time of production, the more unpredictable the capacity becomes (H.-P. Wiendahl 1997, p. 13). This time-based variance is one of the main reasons for the existence of different planning levels. In this way, a rough estimation of the capacity (usually based on standard values) is used during load planning, while increasingly accurate considerations of the actual capacity are used during scheduling and the creation of the finite production plan (Lödding 2008, pp. 84-86).
- **Material availability check** requires the consideration of several uncertainties: regarding demand, quantity of material to be effectively received (which can differ from the one in the purchase order), and replenishment lead time (for production and purchase orders) (Günther et al. 2016, p. 240). The values associated to these uncertainties need to be properly determined.
- **Lot size determination** influences different entrepreneurial objectives. It concerns the calculation of lot sizes for both, production and purchase orders. The objective is to minimize lot-size-related costs, namely storage, production, and setup costs; while keeping production and storage levels within the prescribed lower and upper bounds (Lödding 2008, p. 88; Coniglio et al. 2018, p. 764).
- **The determination of production logistics-related costs** is mostly directly dependent on the other calculated values (e.g. the higher the stocks, the higher the inventory costs). Relevant is, however, the estimation of externally influenced factors such as, among others, material costs, financial effects of delays, energy prices, etc. Methods of cost calculation are utilized for price determination, which in turn affects the production strategy (e.g. prioritization of orders).
- **Load planning and sequence planning** make use of all of the above-mentioned variables, either as standard or as estimated values (depending on the required level of detail). They apply them to their optimization tasks, in which they also ponder the goals and objectives of each enterprise. Additional sources of uncertainty are related to changes in customer orders (specifications, quantity, dates) and urgent orders (H.-H. Wiendahl 2002, pp. 27-30).



---

Additionally, PPC systems can profit from algorithms and models concerning:

- Quality: internally (scrap and rework) and externally (in received goods) originated quality issues can affect PPC plans and their execution.
- Maintenance: in its three forms – corrective, preventive, and predictive – it affects the PPC plans and their execution through planned and unplanned breakdowns.

Quality and maintenance depend on a large number of factors. This leads to the creation and utilization of complex algorithms and models.

The clear separation of the mentioned values and their determination is merely for the purpose of understanding them. In reality, all of them are deeply intertwined. Either as a simple dependence (e.g. throughput time and capacity), as a hierarchy (elements of throughput time in funnel model), or as chained steps (material availability check and capacity calculation before sequencing).

Many values are not estimated each time they are required. Instead, the determination is made on a regular basis (with different frequencies). It is, for example, the case of throughput time and capacity, which are stored in the material and the work station master data, respectively. Although even standardized production environments have to deal with the inaccuracies of this approach, the time and effort (computing resources) involved in carrying out these calculations justify its application.

Algorithms and models dealing with forecasting problems face a basic issue: every forecast is essentially false. What matters is to which degree they are false (Günther et al. 2016, p. 219).

Furthermore, model-based approaches suffer from inaccuracies when faced with complex reality due to the abstraction (reduction and simplification) in their construction. Moreover, they have difficulties taking into account nonlinear relations between inventory, utilization, and performance (Westkämper et al. 2000, pp. 847-865).

The utilization of previously determined values does nothing else but increasing the level of these inaccuracies. In order to deal with them, PPC systems use buffers. Although they lower the effect of stochastic factors, they also mean increasing time, money and space

requirements. Enterprises are therefore forced to find a balance between the risks they are willing to take and how many resource they can dedicate to using buffers.

In addition to the negative effects of buffers, their positive effects are also limited. Westkämper et al. (2000, p. 845) refers to the fact that often the buffers are no longer available anymore as soon as the process begins.

Such inaccuracies in planning are usually handled incorrectly during the execution in the corresponding control activities. The so-called *vicious manufacturing cycle* originates when trying to correct problems (delays) on particular orders while ignoring the behavior of and the effects on the whole system (Mather et al. 1977, pp. 27-51; Westkämper et al. 2000, pp. 844-845; H.-H. Wiendahl 2012, p. 403).

Mass Personalization, as stated before, contributes to increasing the complexity of such already complex problems. Furthermore, the demand for short delivery times and low prices intensifies the pressure (in a production system already stressed by the required flexibility) (Piller 2008, p. 141). This also reduces the possibility to use buffers (e.g. in throughput time, stocks, etc.) as they would increase times and/or costs to an unaffordable extent.

In order for internal complexity to match the external one, two actions are required. On the one side, it is necessary to construct on-demand manufacturing systems. These are characterized by flexibility and changeability (ElMaraghy et al. 2009, pp. 3-24; Hu 2013, p. 6).

On the other side, order management systems need to be able to keep up with this development. H.-H. Wiendahl (2009, pp. 197-212) describes the necessity for PPC systems to be able to adapt to the speed of change. He also identifies the effects that such complex market conditions can have on PPC systems, calling them *turbulences* (H.-H. Wiendahl 2007, pp. 443-446). He labels the causes for these turbulences as *germs* and divides them into five categories: variations (heterogeneous requirements within the same period), fluctuations (heterogeneous requirements within different periods), plan adaptations (because of frequently changing circumstances), deviations (unexpected

events after order release), and inconsistent tolerances (between sourcing, production, and delivery). These are further explained in Figure 2-8.

Cluster	Source	Make	Deliver
Variation	Heterog. sourcing conditions • Times • Quantities	Heterog. order flow • Batch creation • Heterogeneous order mix	Heterog. delivery requirements • Times • Quantities
Fluctuation	Fluctuations in sourcing • Times • Quantities	Fluctuations in production • Fluctuating order mix • Fluctuating availability	Fluctuations in demand • Times • Quantities
Adaptation	Sourcing adaptations • Sourcing deviations • Product cycles (sourcing)	Production adaptations • Capacity demand deviations • Technology cycles (production)	Demand adaptations • Forecast deviations • Product cycles (final products)
Inconsistency	Sourcing tolerances ↔ Production tolerances ↔ Delivery tolerances		
Deviation	Material deviations (sourcing)	Quality deviations (production)	Changes to final product
	Unreliable suppliers • Deadline • Quantity	Unexpected stoppages • Machine breakdown • Staff shortage	Order changes • Deadline • Quantity

**Figure 2-8: Morphology of turbulence germs in production planning and control (H.-H. Wiendahl 2007, p. 444)**

In Mass Personalization, master data either become rapidly invalid (e.g. capacity) or are unavailable (e.g. material data, routings, bills of material, etc.). This affects not only the forecasting but also the decision aspect: without experience, it is difficult to assess the effects of decisions.

All of these factors require a more frequent application of algorithms and models to assist the forecasting and decision-making processes. These, in turn, need to be able to deal with the emerging internal (in response to the external) complexity: the production of an unknown number of product varieties with equally unknown effects on the source, make, and deliver phases.

New IT tools and components are necessary for PPC systems to deal with these arising requirements and the existing challenges. However, they face another issue: although IT systems play an increasingly important role in enterprises, they also require effort and expenses (in many cases considerable). Furthermore, many companies already use IT solutions for order management and related tasks. Some even have many, as the

functionalities are usually distributed among several IT solutions. Therefore, enterprises will try to increase the efficiency of their IT landscape by reducing the associated expenses and increasing the utilization of existing systems (Dürr 2013, p. 5). Additionally, the work in turbulent and evolving markets means that enterprises are required to constantly adapt their IT solutions (H.-H. Wiendahl 2009, pp. 197-198), with the expected effect on the desired IT efficiency. Ideally, IT tools and components should be able to flexibly adapt to changes and, when necessary, complement existing solutions.

To sum up, the following points are relevant for characterizing Mass Personalization and deriving requirements to a solution:

- PPC functions are implemented through interdependent algorithms and models, whose already existing limitations are further put under stress by the requirements of Mass Personalization.
- PPC functions can work with different levels of accuracy (e.g. through estimations) and consider different factors (directly and indirectly influencing them), depending on their context and expected efficacy.
- PPC functions must face turbulences from the production environment (affecting the disposition and deliver processes), which are further increased by Mass Personalization.
- With Mass Personalization, normally little to no master data or experience is available while executing the PPC functions.
- Flexible and changeable manufacturing systems are necessary to deal with the requirements of Mass Personalization, influencing the need for adequate IT tools.
- Any approach should take into account aspects of IT efficiency and the complementation of existing solutions.

### 2.4.3 The Issue of Data Quality

Missing and false data affect the accuracy of planning and control processes negatively and, by extension, the forecasting and decision-making aspects (Westkämper et al. 2000,

p. 844; Günther et al. 2016, p. 240; Reuter et al. 2016, pp. 545-546; Schuh et al. 2017, pp. 425-426). This is worsened by the application of planning with a strong deterministic character and by the presence of complex control loops with a short expected reaction time (Kletti 2007, p. 22). H.-H. Wiendahl et al. (2005, pp. 642-646) identifies two stumbling blocks directly related to data quality: errors in PPC parameters and insufficient quality of feedback data.

In spite of their importance, many PPC systems in enterprises suffer from the low quality of their transaction and master data (Reuter et al. 2016, p. 546). Taking into account that, in many cases, master data can be derived from transaction data, the problem of data collection becomes even more prominent.

Günther et al. (2019, p. 585) identifies – based on the concepts of Goos et al. (2002, pp. 30-32), Wang et al. (1996, pp. 31-32), Wand et al. (1996, pp. 93-94), Price et al. (2016, pp. 241-242), and Batini et al. (2009, pp. 6-9) – five main data quality dimensions relevant for PPC systems:

- **Accuracy:** Are the data correct and reliable?
- **Completeness:** Are the data that represent important states of the reality available?
- **Consistency:** Do the data present the same format? Are they compatible with previous data?
- **Timeliness:** Does the age of the data and/or the time of their availability correspond with the intended usage?
- **Relevance:** Are the data relevant for the intended usage?

Despite the growing digitalization of production environments, much of the feedback activities is still performed manually (Günther et al. 2019, p. 589). Furthermore, the utilization of automatized data collection systems is no guarantee for good data quality (Reuter et al. 2017, p. 489).

PPC systems also face the problem of usually having to function in a heterogeneous software landscape. Lack of integration between the solutions and devices available is another cause of low data quality (Rudolf et al. 2015, pp. 13-16).

Given the already addressed increase in the need for algorithms and models due to Mass Personalization, it is understandable that the demand for data will increase accordingly. IT solutions should take the problem of poor data quality into account and be able to deal with it in the best possible manner.

To sum up, the following points are relevant for characterizing Mass Personalization and deriving requirements to a solution:

- Under Mass Personalization the demand for data is prone to increase.
- The quality of master and transaction data is frequently not adequate for the intended use.

#### 2.4.4 Summary

Production management systems already have to deal with uncertainties to achieve the set objectives in the best possible way. Mass Personalization and its corresponding turbulences only contribute to making their tasks more difficult. Although there are similarities with other manufacturing strategies, the production of a large number of products with freely configurable features forces production management systems to forecast and make decisions with little to none experience. This pressure is transferred to the assisting software solutions and their functionalities. They do not only have to provide responses under situations far more complex than before and be able to continuously adapt; they have to do it in the most efficient way possible.

Table 2-3 summarizes the characteristics of Mass Personalization and the derived requirements for a production management software that is able to deal with them. For this, the basic conditions and elements presented in this chapter are taken into account.

The characteristics of Mass Personalization (with the corresponding section in parenthesis) are:

- Personalization (free definition) of final products and their components (2.1.1, 2.1.2)
- Unknown demand of final products (in kind and quantity) (2.1.2)

- Expected similar price as in mass produced products (2.1.2)
- Expected similar delivery time as in mass produced products (2.1.2)
- High demand (quantity) similar to mass production (2.1.2)
- Increasingly turbulent production environment (2.4.2)
- Lot size one production (2.1.2)
- Unknown demand of raw and semi-finished goods (in kind and quantity) (2.1.2)
- Limited possession of master data (2.4.2)
- Algorithms and models to assist forecasting and decision-making are more demanded and must be more capable (reduced buffers) (2.4.2)
- Increase in demand for data (2.4.3)
- Lack of experience in producing the specified product or component (2.4.2)
- Utilization of flexible and changeable manufacturing systems, with corresponding management systems (2.4.2)

Consequently, an adequate production management software is required to

- be able to deal with an unknown high demand (quantity) of unique products;
- be able to deal with a high amount of freely configurable product features;
- be able to deal with changing and unknown disposition and deliver situations;
- be able to consider direct and indirect influencing factors and effects;
- be able to deal with low quality, non-existing, and not acquirable data;
- be able to deal with related and interdepending algorithms and models, providing different levels of detail;
- be able to provide an adequate response in the required time;
- be able to adapt to and extend existing production management systems; and
- enhance the IT efficiency.

Instead of PPC or order management, the slightly more general term *production management* is used. This is because the relevant functions are usually distributed among several software solutions.

Existing enterprises moving into Mass Personalization will most probably employ some sort of software solution. The integration in and expansion of an existing IT landscape (that in many cases can be considerably complex) must therefore been taken into account.

All possible solutions will require data. Additionally to low data quality, two issues gain importance:

- non-existing data corresponding to the production of unknown personalized products and components (e.g. bill of material, processing time, etc.)
- non-acquirable data corresponding to the production of single products or components, which affects the learning effect negatively

In spite of the importance of data, new methods for data collection are not a part of this work.







# 3 Data Analytics to Assist Production Management Systems

This chapter introduces the concepts of data analytics approaches relevant for their application in assisting production logistics tasks. The intention is not only to explain their advantages, but to point out how they can be utilized and what their limitations are. Therefore, the purpose of this work is not to describe in detail any technique, but to consider the general aspects of data analytics.

The objective of this chapter is to identify the features of a possible data analytics-based solution that addresses the requirements derived from Mass Personalization to a production management software.

## 3.1 Basic Concepts of Data Analytics

The last few years have seen a great enthusiasm regarding the utilization of tools for Big Data, machine learning, etc. At the same time, this has brought about a confusion regarding the meaning and application of the approaches available. In this regard, manufacturing has been no exception.

This subchapter will introduce the basic concepts necessary for the usage of data analytics (relevant within the scope of this work).

### 3.1.1 Data Analytics-based Approaches

The first important concept to be specified is that of **data mining**. Hand et al. (2001, p. 1) defines it as “the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and

useful to the data owner". Witten et al. (2017, p. 6) adds that this process is "automatic or (more usually) semiautomatic".

A related concept is that of **statistics**. This kind of applied science makes use of the probability theory (its mathematical foundation) to analyze and model data (Bruce et al. 2017, p. 1). Two types of statistics can be distinguished: descriptive, "summarizing the data in a convenient and concise way"; or inferential, "allowing one to make some statement about the population from which the data were drawn or about likely future data values", based on samples (Hand et al. 2001, p. 166; Cleff 2015, p. 4).

Within data analytics, data mining focuses on discovering new information from data; an adventurous process, as there are no expectations given on the result (ISO/IEC 13249-6:2006, p. 6). It therefore differs from statistics, which focuses on collecting data to answer specific questions. There are two other big differences: one is the size of the data, as data mining is adequate for large-size data sets (problematic for classical statistical approaches). The other is the nature of the data analyzed, as data mining deals with heterogeneous data usually generated for other purposes (secondary analysis), while the data processed in statistics have normally been created for that specific usage (primary analysis) (Hand et al. 2001, p. 2).

At the same time, statistical methods and ideas are fundamental for data mining and it extends them in order to apply them to situations with large and complex data (Hand et al. 2001, pp. 18-19). Both concepts make use of two elements: the mathematical model (which constitute the base for both approaches) and the computational algorithm (essential for data mining and almost a requirement for modern application of statistics) (Hand 1999, p. XXVIII).

Data mining comprises four types of techniques (ISO/IEC 13249-6:2006, p. 11):

- **Association rules discovery** concentrates on finding rules of the type "if a transaction type contains item X and item Y, then the transaction type also contains item Z in N% of all transactions of this type".
- **Clustering** (segmentation) finds rows with similar characteristics and organizes them into well-defined clusters.

- **Classification** comprises a set of rows with a set of fields and a special field called class label, and tries to compute a model that is able to predict the class label using the remaining fields.
- **Regression** is similar to classification. They differ in the type value to be predicted, with regression concentrating on continuous values.

Data mining is based on the use of algorithms – in contrast to statistics, where the “model is king” (Hand 1999, p. 17). An algorithm is defined in this context as “a well-defined procedure that takes data as input and produces output in the form of models or patterns” and is constituted by five components (Hand et al. 2001, pp. 141-142):

- the **data mining task** (e.g. visualization, regression, etc.)
- the **structure** (functional form) of the model or pattern to be fitted to the data
- the **score function** used to evaluate the quality of the fitting of the model or pattern to the data (e.g. squared error)
- the **search or optimization method** used to optimize the score function (by determining the structure and values to maximize or minimize it, depending on the context)
- the **data management function** used for storing, indexing, and retrieving the data

Hand et al. (2001, pp. 165-166) defines a **model** as “an abstract representation of a real-world process” and “a high-level, global description of a data set” that may be used for descriptive or inferential purposes.

He defines a **pattern**, on the other hand, as “a local feature of the data” that is usually of interest to find “departures from the general run of the data” (e.g. high correlations between variables, a set of items with exceptionally high values, etc.). The reasons for using it may also be descriptive or inferential.

Based on these definitions and the work of Kraker (Kraker et al. (2013) and Kraker (2013), referencing a lecture by Markić), patterns can be understood as the specification of models in order to apply them to gain knowledge. Witten et al. (2017, p. 6) is in the same line of argumentation by defining the pattern discovery as the objective of data mining.

This work will mainly refer to models, taking into account the existence of different degrees of specification. Patterns will be addressed as the element used to interpret the results of the analytical process.

Three phases are distinguished when employing data mining techniques (ISO/IEC 13249-6:2006, pp. 11-13):

- In the **training phase**, the model is computed based on the data and the corresponding settings.
- In the **test phase**, the quality of the computed model (its prediction capability) is checked using a test data set.
- In the **application phase**, the computed model is utilized for its intended purpose, applying it to the corresponding data set.

Not all techniques require all three phases – for example, it does not make sense testing the created clusters. Figure 3-1 sums up the inputs and outputs of each phase, how they are related, and which technique each phase requires.

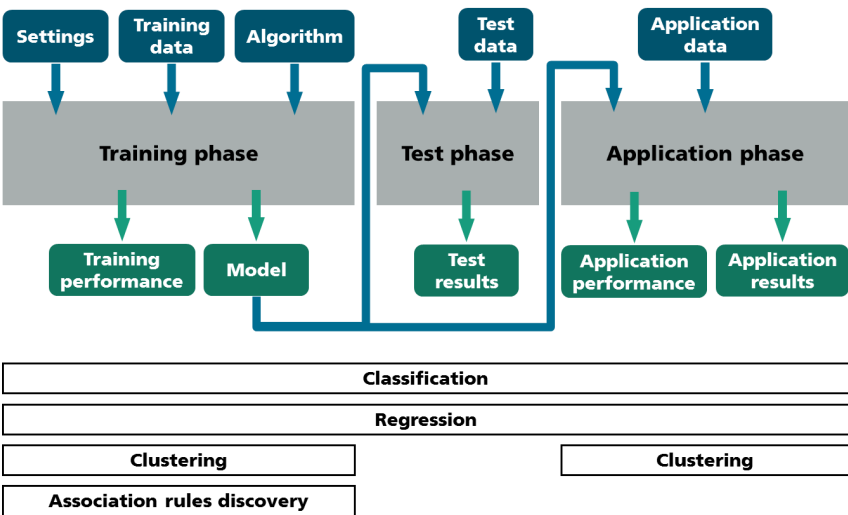


Figure 3-1: Phases for data mining techniques

The phases take place at different moments, with different running times and resource requirements.

Of special interest is the concept of **Machine Learning**. Samuel (1959, p. 210) refers to machine learning when affirming that “programming computers to learn from experience should eventually eliminate the need for [...] detailed programming effort”. Alpaydin (2016, pp. 14-25) refers to it as a way of learning by extracting rules from data. The learning process is based on a learning algorithm that adjusts the parameters of a model in order to optimize a defined performance criterion.

Three main types of machine learning approaches can be distinguished:

- In **supervised learning**, input and output are clear. The aim is “to learn a mapping from the input to the output” and a “supervisor provides the correct values” (Alpaydin 2016, p. 111). A typical example is classification.
- In **unsupervised learning** “there is no predefined output and hence no [...] supervisor”. Its aim is to “find the regularities in the input” (Alpaydin 2016, p. 111), or in other words, a structure (Alpaydin 2016, p. 117). A typical example is clustering.
- **Semisupervised learning** deals with situations between the two previous approaches. While the input for supervised learning consists of labeled data (the content is described and interpreted) and unsupervised learning usually deals with unlabeled data (without a described content), the goal of semisupervised learning is to perform classification (usually only possible with labeled data) by combining unlabeled and labeled data (Witten et al. 2017, p. 468).

Sutton et al. (2018, pp. 1-9) considers an additional type: **reinforcement learning**. It consists of an iterative process constituted by a “learning agent interacting with its environment to achieve a goal”. It is composed of the learning agent, the policy for controlling its behavior, the environment (with its corresponding state), the reward signal indicating the effect on the environment of the actions of the learning agent, a value function to predict future rewards, and, optionally, a model of the environment. The methods try out different alternatives, exploiting old actions and exploring new ones, in order to maximize the reward signal. While approaches of this type present more

characteristics than the ones mentioned, this work will concentrate mainly on their iterative and interactive character.

One can also differentiate the methods depending on when the learning takes place: *eager* methods produce a “generalization as soon as the data has been seen” (Witten et al. 2017, p. 85), while *lazy* methods differ the real learning process beyond training until a new instance (case) must be classified (Mitchell 1997, p. 244). The latter are referred to as **instance-based learning**. With such methods, learning during the training phase merely comprises storing (memorizing) the instances found (with some kind of representation). When a new instance is encountered, the stored instances are accessed and compared to the new one in order to classify it (Mitchell 1997, pp. 230-231; Witten et al. 2017, pp. 84-85).

Figure 3-2 provides an overview over the mentioned machine learning approaches.

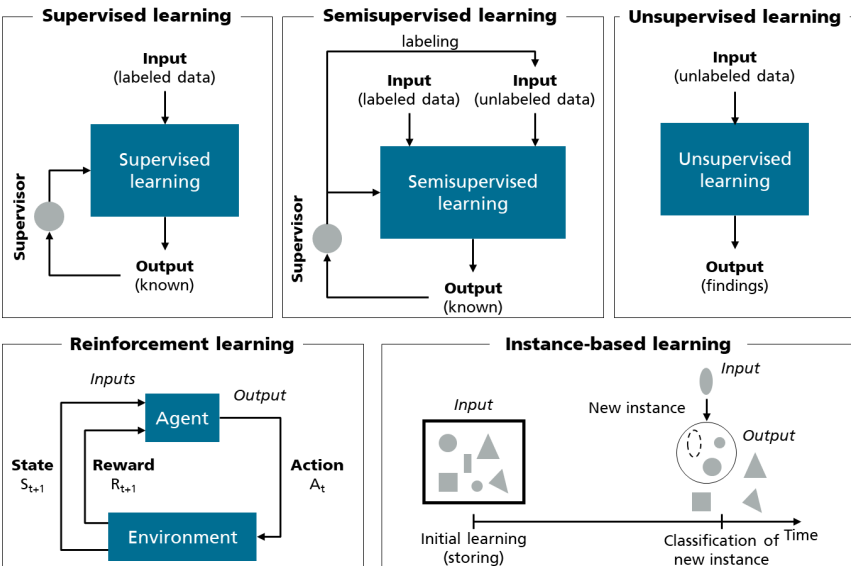


Figure 3-2: Overview over machine learning approaches

(based on Jones 2017; Sutton et al. 2018, p. 54; and Witten et al. 2017, pp. 468-472)



Although machine learning and data mining are considered separately, mainly because of the philosophy of the former based on a self-learning approach, they are deeply intertwined. This is up to the point where established data mining techniques originate in machine learning (e.g. classification, clustering, etc.), making the boundaries between them unclear (Fayyad et al. 1996, p. 43; Hand et al. 2001, p. 4; Witten et al. 2017, pp. 44-46). Therefore, this work will mainly refer to data mining when considering aspects that affect both approaches.

Approaches that make use of algorithms to generate models based on the data (data mining, machine learning) can be referred to as **data-driven**, whereas classically those based on the direct application of models using stochastic assumptions (statistics) are known as **model-based** (Breiman 2001, p. 199; Freitag et al. 2015, p. 24). The latter concept can be extended through the consideration of transfer learning (see section 3.2.2).

As can be seen, many concepts come into play when dealing with deriving information from data. To address all of them, the term **data analytics** (DA) can be used. Runkler (2016, p. 2) defines it as “the application of computer systems to the analysis of large data sets for the support of decisions” while referring especially to its interdisciplinary character, as it encompasses concepts from other disciplines (such as statistics, machine learning, artificial intelligences, etc.). Ridge (2015, p. 4) provides a more practical approach, referring to data analytics as “any activity that involves applying an analytical process to data to derive insights from data”. Data analytics can then be seen as a general term that comprises all analytical activities.

Analytics can be organized into four types (Rollings et al. 2017, p. 8; Gartner 2019):

- **Descriptive analytics** deals with the question “What happened?” (or “What is happening?”).
- **Diagnostic analytics** tries to answer the question “Why something happened?” (or “Why is it happening?”).
- **Predictive analytics** addresses the question “What will happen?”

- **Prescriptive analytics** concentrates on the question “What should be done to make something happen?”

Although, at first glance, it would seem as if the types were sorted in order of ascending development degree, the techniques utilized in each type depend on the complexity of the situation at hand (Colangelo et al. 2016b, p. 277).

It is common to combine several analytical approaches to answer questions or solve problems. This is referred to as **advanced analytics** (Bose 2009, pp. 155-156). Prescriptive analytics constitutes a good example. It normally comprises elements of operations research: optimization, simulation, and evaluation methods; all of which are combined with predictive analytics in order to assist during decision-making and optimization (Liberatore et al. 2011, p. 582; Evans et al. 2012, p. 5; Schniederjans 2015, p. 120; Soltanpoor et al. 2016, p. 247; Bertsimas et al. 2020, pp. 1025-1043).

This work will mainly refer to data analytics to describe the intended solution. The main focus will be on the areas of data mining and machine learning, since they provide innovative approaches necessary for the issues presented in chapter 2 (as to be explained later). The associated area of statistics will also be taken into account as it supports and complements both methods and all four types of data analytics.

Accordingly, this work will use the term **data analytics-based** when referring indistinctly to data-driven and model-based approaches.

Approaches of operations research (among others, simulation), which also play a role (primarily in prescriptive analytics), will be considered as black boxes, as their specific requirements are not within the scope of this work. Such methods are also based on mathematical modelling and probability, and are usually implemented using computational algorithms (Taha 2017, pp. 34-38).

### 3.1.2 Complementing Concepts

Additionally to the reviewed terms it is necessary to consider some commonly used concepts. One of great importance in the entrepreneurial context is **Business Intelligence** (BI). This form of data analytics refers to the employment of various tools and applications to create key figures that are utilized to assist diverse administrative tasks (Wierse et al. 2017, p. 35). Assisting decision-making processes, performance measuring (e.g. OEE), or serving as a basis for continuous improvement processes (CIP) are some application examples (Dedić et al. 2016, p. 225; Wierse et al. 2017, p. 35).

One of the concepts often creating a big confusion is **Big Data**. There are numerous definitions, many of which tend to focus on particular aspects relevant for the work area of the corresponding author (Dedić et al. 2017, p. 116). One basic and commonly accepted definition is that by Gartner (2019): “Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision-making”.

According to this definition, Big Data can be viewed as problem (complex data processing) and as opportunity (possibly valuable information assets). It is based on the so-called 3Vs (Furht et al. 2016, p. 3; Oussous et al. 2018, p. 433):

- **Volume** refers to the size of the data and to the related structures (records, transactions, tables, etc.).
- **Velocity** refers to how rapidly the data is generated and transferred (batch, near time, real time, and streams), and the rate at which it should be analyzed.
- **Variety** refers to the multiplicity of formats in which the data is generated because of the diversity of distributed sources.

Over time, additional Vs have been added. These are: veracity, which refers to how reliable the data are; and value, which refers to the usability of the data (Gandomi et al. 2015, p. 139; Kacfeh Emani et al. 2015, p. 72; Oussous et al. 2018, p. 433).

With regard to variety, three types of data should be considered (Gandomi et al. 2015, p. 138):

- **Structured data** refers mainly to tabular data found in spreadsheets or relational databases. Its processing can be automatized without considerable effort.
- **Unstructured data** refers to data that do not possess a structure that allows for a simple automation of its analysis. For example, texts, video, audio, etc.
- **Semi-structured data** refers to the midpoint between the two types; when the data possess a structure that is not standardized.

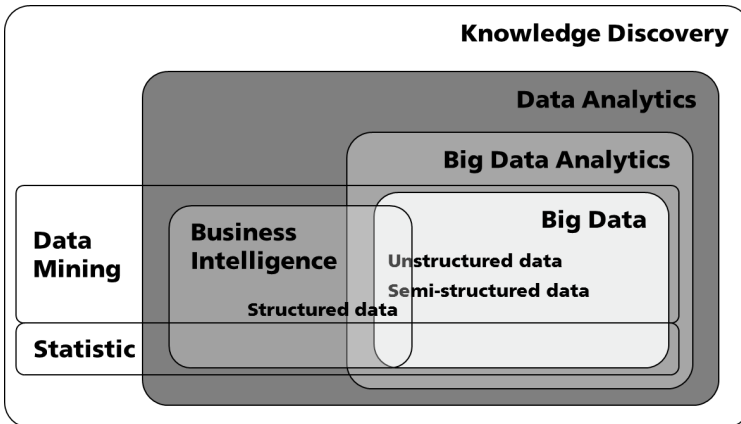


Figure 3-3: Overview over concepts of data analytics (based on Dedić et al. 2017, p. 115)

The term **Big Data Analytics** (BDA) is used to describe analytics tools specifically developed to deal with the requirements corresponding to Big Data, namely the analysis of large volumes of disparate, structured, and unstructured data (Chan 2013, p. 8; Belle et al. 2015). However, they are still a form of data analytics (George et al. 2014, pp. 323-324; Dedić et al. 2017, p. 117).

Figure 3-3 provides an overview over these concepts.

## 3.1.3 Knowledge Discovery in Databases

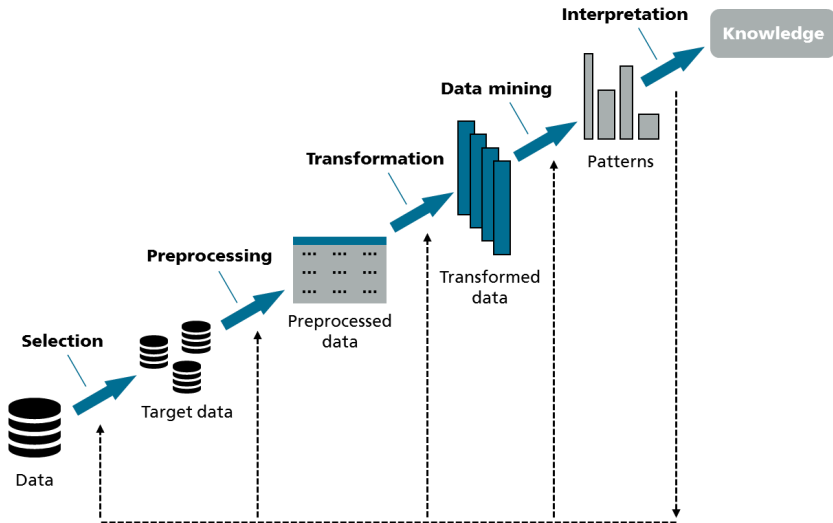


Figure 3-4: Knowledge Discovery in Databases (Fayyad et al. 1996, p. 41)

Far from being standalone tasks, the derivation of insights from data must be understood and performed as a process in order to succeed. According to Fayyad et al. (1996, pp. 39-41), the term **Knowledge Discovery in Database** (KDD) “refers to the overall process of discovering useful knowledge from data”, being knowledge “purely user oriented and domain specific and [...] determined by whatever functions and thresholds the user chooses”. This knowledge should be valid, novel, and potentially useful. The process is illustrated in Figure 3-4.

Every step identified by Fayyad et al. (1996, pp. 41-43) – based on the work by Brachman et al. (1996, pp. 37-58) – has a specific task. The first task (not depicted in the illustration) is understanding the application domain and identifying the objective of the KDD process from the customers’ point of view (e.g. the enterprise). The task of each step is described as follows:

- **Selection** is the step in which, based on the understanding of the data and the objectives, a determination of the relevant data is performed (Ester et al. 2000, p. 2). The result is the *target data*.
- **Preprocessing** consists of two main parts. The integration takes care of bringing together data from various sources, usually with different formats and structures. Data cleaning tries to solve the data-quality-related problems, like the ones introduced in section 2.4.4. The result of this step is *preprocessed data* (Ester et al. 2000, p. 3).
- **Transformation** is the step in which the data is transformed to an appropriate and usable representation (Ester et al. 2000, p. 3). Typical transformations are (Ester et al. 2000, pp. 3-4; Witten et al. 2017, pp. 288-315):
  - attribute selection (or feature selection), in which the most relevant attributes for the subsequent analysis are chosen, either as an assessment (filter method) or directly using the intended analysis (wrapper method), and performed manually or automatically;
  - discretizing, in order to obtain nominal attributes;
  - projection, in which general mathematical transformations are performed, and
  - sampling, in which an adequate sample out of a large volume of data is extracted.The result of this step is *transformed data*.
- **Data Mining** is the step in which an analysis per se is performed, applying algorithms and models. Four parts can be recognized in this step: (1) selection of the method(s) to be used (which must match the goal of the KDD process) and, if necessary, the creation of the corresponding algorithm/model; (2) model fitting, in which the parameters are adjusted; (3) evaluation, in which the model is evaluated (tested); and (4) model refinement, referring to the iterative refinement of the model (Brachman et al. 1996, p. 44; Fayyad et al. 1996, p. 42) . The result of this step are the *patterns*.
- **Interpretation** is the step in which the gained patterns and values are evaluated and the corresponding actions are triggered: “using the knowledge directly, incorporating the knowledge into another system for further action, or simply documenting it and reporting it to interested parties” (Fayyad et al. 1996, p. 42).

---

The process is iterative within the steps and between them, providing for every step the possibility to go back to any of the previous ones (Fayyad et al. 1996, p. 42). Going back and forward in the process is not at all unusual, as often several cycles are necessary in order to achieve the desired result (e.g. selection of new data sources, features, data mining methods, etc.).

The evaluation in the data mining step is key for the process. It allows analyzing the performance of a chosen method (algorithm/model). This can trigger different actions: changing the parameters, choosing another data set for training and testing, or changing the method (as it enables the comparison between different ones). Statistical techniques are used to perform the evaluation (Witten et al. 2017, pp. 161-202).

Essential is the correct selection and utilization of training and test data sets. This not only requires the correct generation of samples. Advanced techniques allow for a dynamic employment of the data sets. An example is *repeated holdout*, which consists of iterative training and testing interchanging the data sets in order to reduce bias in the estimation of the error rate. The data sets can be partitioned using a technique known as *cross-validation*. With sufficient data, it is even possible to split the training data sets into smaller ones in order to test different parameters (Witten et al. 2017, pp. 167-172).

Sampling for the data sets is a relatively simple procedure that makes use of methods of mathematics and probability in order to improve the standard random sample. For example, stratification allows for equal representation of classes in training and test data sets (Witten et al. 2017, 167, 315, 368-369).

When evaluating predictions, often the corresponding costs of making a wrong assumption can be considered. Mathematical methods (confusion matrix, cost curves) can be used to analyze the costs of the different cases (false positives and false negatives) (Wierse et al. 2017, pp. 62-63; Witten et al. 2017, pp. 179-182). Accordingly, it is possible to consider the cost factors during the learning process in order to improve it (Witten et al. 2017, p. 183).

Tsai et al. (2015, p. 3) proposes a simplification and generalization of the KDD process. He does this by organizing the steps into three phases: input, data analysis, and output (Figure 3-5).

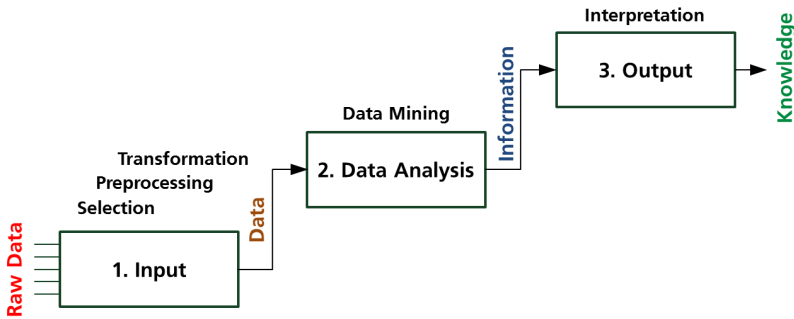


Figure 3-5: Simplification of the KDD process (Tsai et al. 2015, p. 3)

Tsai et al. (2015, p. 4) argues that other techniques besides data mining are part of the KDD process (e.g. statistics). Brachman et al. (1996, p. 44) also speaks of data analysis to describe the information-gaining phase. In line with this thought and based on the definitions in section 3.1.1 this work will use the comprehensive term **data analytics** when referring to the tasks of data analysis.

Data analytics approaches do not only belong to the data analysis phase, they are also applicable during the input phase. For example, it is usual to apply data mining techniques to assist during the preprocessing and transformation steps (Witten et al. 2017, pp. 304-331).

A KDD process can require the application of several types of data analytics (algorithms/models). Either in different steps (input, data analysis) or within the same step (e.g. creation of clusters which are later used in classification).

Also applicable during the input phase is **explorative data analysis** (EDA). Seltman (2018, p. 61) refers to it as “any method of looking at data that does not include formal statistical modelling and inference”. It concentrates on trying to understand the data,



detecting problems, determining relationships among the variables, and making preliminary assumptions. It can be considered as an approach making a joint application of descriptive and diagnostic analytics (Freitag et al. 2015, p. 24).

It is of importance to facilitate the interpretation of the information (patterns) generated. Whenever the human factor is involved, visualization plays an important role in providing cognitive support (Tory et al. 2004, p. 73). It assists not only in the specific output phase (which can have different degrees of automation) but also in the other steps – for example, during the EDA or the evaluation of methods. The term **Visual Analytics** describes technologies based predominantly on visual representations and the facilitation of user interaction (Wong et al. 2004, p. 20). This is also one of the cornerstones of BI (Obeidat et al. 2015, pp. 52-53).

There are other processes comparable to KDD such as SEMMA (Sample, Explore, Modify, Model, Assess) and CRISP-DM (CRoss-Industry Standard Process for Data Mining). Although the steps are described differently to accentuate the focus of the method (e.g. CRIPS-DM emphasizes the understanding), they can be considered as implementations of KDD (Azevedo et al. 2008, pp. 184-185).

To sum up, the following points are relevant for the determination of features of a possible data analytics-based solution:

- A great number of commonly used terms (e.g. data mining, Business Intelligence) can be grouped under the concept of data analytics and the shared objective of knowledge discovery.
- Data analytics can be implemented using a variety of approaches (of both types, data-driven and model-based).
- Data-driven approaches allow for the dynamic generation of models based on algorithms in order to create solutions that fit an specific context.
- Data analytics approaches are to be considered as processes, either regarding their own execution form (e.g. instance-based learning), the general data mining phases, or the knowledge discovery process; and the corresponding steps can be deferred in time.

## 3.2 Data Analytics in Practice

Due to the complexity of some approaches, the development of data analytics techniques requires a great amount of work on the theoretical side. However, in order to be usable, these need to be implemented in a practical context. Relevant elements and characteristics of this process are explained in this section.

### 3.2.1 Challenges

Despite their many advantages, data analytics approaches also face challenges when applied to real world problems.

*Occam's razor* refers to the conflict between accuracy and simplicity (interpretability) (Breiman 2001, p. 206). In this context, accuracy refers to reducing both, the bias error, introduced by being unable of "capturing the underlying model"; and the variance error, "due to the sensitivity to noise in the data". The complexity of a model depends on the "model type, number of inputs and number of parameters" (Lever et al. 2016, p. 703).

When several methods are possible, the one fitting the data best is selected (Witten et al. 2017, p. 34). Accurate models can rapidly become complex and unintelligible (especially true in the case of machine learning). This will decrease the bias but also increase the variance (overfitting). On the other hand, models that are too simple will have low variance and high bias (underfitting).

The main issue with data analytics projects is their costs. The discovery-based, iterative nature of the process involving many resources and (expensive) specialists requires considerable amounts of effort (time and money) (Bose 2009, p. 164; PAC 2014, p. 25; Gröger 2018, p. 9). The following cost elements can be considered (based on Marbán et al. 2008, pp. 135-137 and Wierse et al. 2017, pp. 356-383):

- **infrastructure:** acquisition and installation of software and hardware

- **implementation:** execution of iterative phases of data input, analysis, and output (including realization in the software tools); integration (data interfaces); commissioning; and other related tasks (e.g. staff training)
- **maintenance:** correction of errors and update of algorithms/models, general maintenance of software tools and hardware

It is important to distinguish between investments (one-time accruing costs), such as the purchase of hardware and software, or the development of algorithms/models; and operation costs (continuously accruing), such as support or licenses. The utility of the investments requires time in order to be evaluated (Dürr 2013, p. 98). If several analytical solutions are required at the same time, the investment and some costs (e.g. data input) can be better utilized (Wierse et al. 2017, p. 401).

The phase that by far requires the largest part of the budget is that of data input (often referred to as *data preparation*). 80% of the effort goes into this phase (Stonebraker et al. 2018, p. 8).

Data analytics techniques (and specially data mining) are usually associated with high computational costs (Al-Jarrah et al. 2015, p. 87). They are then performed asynchronously (in batch mode) or synchronously (in online mode) (Sayad 2011, p. 12). In both cases, the running time must match the requirements (Han et al. 2012, p. 31).

Concepts like *real time data mining*, *online learning* and *incremental learning* usually describe – though not meaning exactly the same – situations where the analytical solution needs to learn from rapidly changing data streams, unlike approaches where learning is done in batches (Sayad 2011, pp. 12-14; Witten et al. 2017, pp. 509-512). Constant learning is achieved in a number of ways – for example, by updating parameters, creating mini batches, etc. – in order to adapt to new instances. This type of approach becomes relevant for rapidly changing and time critical scenarios, such as robotic and autonomous driving (Gepperth et al. 2016, pp. 357-361). As such scenarios are not common in production logistics (though they can appear in manufacturing, e.g. in machine or process monitoring), the solution proposed in this work will not address the requirements of such approaches directly.

Limitations regarding execution time and computing resources will be further discussed in chapter 4.

To sum up, the following points are relevant for the determination of features of a possible data analytics-based solution:

- When selecting and developing an analytical approach it is necessary to achieve a balance between accuracy and simplicity.
- Approaches to either reduce the investment and operation costs or increase their resulting utility are necessary.
- Data analytics can be performed asynchronously (in batch mode) or synchronously (in online mode).

### 3.2.2 Domain Knowledge

The utilization of **domain knowledge** is important in all phases of the KDD process, as it not only enables (e.g. through understanding of the process) but also accelerates them (e.g. through restriction of alternatives in data cleaning) (Brachman et al. 1996, p. 47; Fayyad 1996, p. 50; Szczuka et al. 2014, p. 343; Witten et al. 2017, pp. 512-515).

For Büchner et al. (1999, pp. 448-449) – based on Anand et al. (1998) – domain knowledge can be utilized for “making patterns more visible, for constraining the search space, for finding more accurate knowledge, and for filtering out uninteresting patterns”. He distinguishes between *objective domain knowledge*, consisting mostly of domain-related facts with little context-dependency (e.g. about a branch) and *subjective domain knowledge*, with a high degree of context-dependency (e.g. about a specific enterprise in a branch or the existing data structures).

On the other hand, the lack of domain knowledge or the knowledge differences of the involved stakeholders can affect projects negatively, increasing the necessary effort. Moreover, developing solutions every time anew leads to elevated costs and long-duration

projects. Reusing analytical solutions could solve this problem (Brodsky et al. 2015, pp. 1418-1419; Gröger 2018, pp. 9-10).

In order to reuse domain knowledge in data analytics, techniques of **transfer learning** can be applied. These function by “transferring knowledge learned in one or more source tasks and using it to improve learning in a related target task” (Torrey et al. 2010, p. 242). This is done in order to improve the performance of the models, accelerate the learning process, and work with data that are either scarce or expensive to acquire (Pan et al. 2010, p. 1345; Torrey et al. 2010, p. 243). Pre-trained models – either created from scratch or acquired off-the-shelf – are applied as starting point in the target task (Shin et al. 2016, pp. 1285-1286). These can contain different types of knowledge to transfer: full models, specific parameters, and relations between data (features). The pre-trained models are subsequently fine-tuned as necessary, a process varying from changing parameters to retraining the model (using the previous structure as a basis) (Pan et al. 2010, pp. 1347-1352). In some cases, the pre-trained model is directly applied (Yosinski et al. 2014, p. 3321). Sutton et al. (2005, pp. 748-750) even proposes combining several separately trained models. Every model is a solution to a simple task, providing the combination of several models a solution to a more complex problem. This approach is also referred to as *hierarchical transfer* (Torrey et al. 2010, p. 247).

As transfer learning relies on the utilization of models – that are fine-tuned as required, if at all – these approaches are also usually referred to as model-based (Aytar et al. 2011, p. 2253; Wang et al. 2019, pp. 367-374).

To sum up, the utilization of domain knowledge and transfer learning is relevant for the determination of features of a possible data analytics-based solution as it allows increasing the accuracy of analytical solutions as well as accelerating their development and reducing the associated efforts.

### 3.2.3 Smart Data

As Fayyad et al. (1996, pp. 41-42) affirms several times, the KDD is no standalone activity: it must be integrated within the processes, from which it receives data and which are passive of the improving actions resulting from the knowledge gained. Recently, the term **smart data** has been used to describe such approaches with the objective of a “data-based design and continuous enhancement of intelligent processes” (Wierse et al. 2017, p. 33). As Wierse et al. (2017) refers not only to data-driven but also to model-based approaches when talking about smart data (though the first ones are predominant), this work will consider *data-based* as equal to *data analytics-based* in this case.

The objective of creating intelligent processes refers to designing them in a way that makes them autonomous and context-aware. While the former is achieved through the utilization of data analytics, the latter requires interaction with the environment, which comprises (as input) historical data, current data, triggers, and (as output) the results of data analytics (Schilit et al. 1994, pp. 85-89; Wierse et al. 2017, pp. 39-45).

Implicit in the concept of smart data is the idea of creating a continuous improvement process based on the data. The continuity not only refers to the cyclic character of the process (see section 2.4.1). It also considers the fact that the work environment is continuously changing and that data analytics-based processes should be able to permanently adapt to these changes (Christensen et al. 2010, pp. 15-16). This continuous learning process will require to steadily control the performance of the models and, if necessary, generate new ones through retraining (even changing the algorithm, if required) (Bang et al. 2019, pp. 116-117).

Wierse et al. (2017, pp. 43-44) mentions the utilization of *case-based reasoning (CBR)* as an example of continuous learning and context awareness. This can be considered as an implementation of instance-based learning in order to solve situations (cases) based on experience (Bichindaritz 2015, pp. 187-188). Aamodt et al. (1994, pp. 45-46) proposes four phases for the CBR cycle:

1. **Retrieve** stored similar cases

2. **Reuse** the retrieved cases to solve the situation (case) at hand
3. **Revise** if the application of the proposed solution was successful
4. **Retain** based on the successful new experiences (new or modified cases)

To sum up, the concept of smart data is relevant for the determination of features of a possible data analytics-based solution as it refers to creating intelligent and continuously improving processes through the integration with the existing ones and their environment.

### 3.3 Assistance of Production Management Systems

This section will explain how data analytics can be applied to assist production management software to address the requirements posed by Mass Personalization. It ends with a summary describing the resulting features of a possible data analytics-based solution.

#### 3.3.1 Application of Data Analytics in Production Management

As explained in chapter 2, production management systems face numerous uncertainties. A situation that is worsened by the requirements of Mass Personalization, taking the capabilities of currently utilized algorithms and models to their limits.

The utilization of techniques of data analytics can greatly assist production management software solutions (Colangelo et al. 2018, pp. 193-194). Important aspects are:

- The high demand of personalized products generates large volumes of heterogeneous data, where relations and causality may be difficult to extract. Dealing with such situations – processing large-size unclear data sets to gain insights – is the strong point of data analytics (especially of data-driven approaches). They allow generating accurate models from the data (albeit sacrificing interpretability). This accuracy is necessary to avoid the need of buffers.

- Their learning capabilities allow data analytics approaches to deal with uncertainties (e.g. future material demand) as well as with unknown situations (e.g. missing master data), being also able to adapt to changes (continuous learning).
- The variety of data analytics techniques and methods allows to work with different levels of detail (according to the requirements and the data available) while delivering an adequate accuracy. Several data analytics approaches can be combined in order to manage complex problems and situations where various processing steps are necessary (e.g. data transformation).
- Domain knowledge transfer allows “outsourcing” the learning process. This allows to apply models with good performance, working in situations with low data quality (e.g. when using a new machine), and accelerating the own learning process.

The forecasting and decision-making aspects can profit from predictive and prescriptive analytics, respectively.

Predictive analytics may be used to address uncertainties (e.g. demand) and estimate performance based on past data (Shao et al. 2014, p. 2195). A case with different levels of details (and of complexity in the prediction) is the estimation of throughput time. It may be predicted directly (Lingitz et al. 2018, pp. 1052-1055) or by calculating the influencing factors – for example, the inter-operation time (Schuh et al. 2018, pp. 169-173) or the capacity, by means of fault prediction (Ji et al. 2017, pp. 188-193).

Prescriptive analytics techniques are able to evaluate what consequences decisions will have on the production system (Shao et al. 2014, p. 2195), thus helping to avoid the vicious manufacturing cycle. The most notorious application of this kind of analytics is for optimization tasks, such as the ones taking place during planning (Heger et al. 2015, pp. 238-244; Priore et al. 2015, pp. 54-58; Kozjek et al. 2018, pp. 209-214). The possibility to work with (and combine) various levels of detail allows for dealing with uncertainties in different planning horizons.

Descriptive and diagnostic analytics also play an important role. The corresponding techniques can either assist the other analytics – for example, using clustering to support scheduling (Chien et al. 2005, p. 328; Tamura et al. 2015, pp. 891-898), or each other



(Brodsky et al. 2015, p. 1419). Descriptive analytics techniques can be used for presenting data summarized in a way that allows understanding what is going on in the production systems (e.g. average throughput time pro product cluster) while diagnostic analytics techniques allow determining cause-effect relationships (e.g. factors that influence downtimes and delays) (Shao et al. 2014, pp. 2194-2195).

As pointed out before, many of the examples reviewed are combining several algorithms/models and techniques, either merged in one or as separate elements, in order to provide the most adequate solution to each situation.

Furthermore, maximal utilization can be achieved through the pursue of smart data-based systems by means of building intelligent processes based on the integration of data analytics-based solutions.

### 3.3.2 Example for Throughput Time Prediction

A typical problem in production management is the prediction of throughput times when planning production orders. Additive manufacturing (also known as 3D printing) allows producing highly personalized individual products; however, the resulting throughput time for each product depends on a number of different factors, making an accurate prediction difficult.

Data documenting previously manufactured products would possess the following attributes: machine (machine ID), main material used (material ID), support material used (material ID), volume of main material (cm<sup>3</sup>), volume of support material (cm<sup>3</sup>), processing time (min), and setup time (min). The considered data table would be the result of merging data from several sources (productions orders and historical data from the 3D printer).

For this example, the utilization of a common classification method known as support vector machines (SVM) is proposed. This supervised learning approach creates, based on the training example, a hyperplane or a set of hyperplanes in a high or infinite dimensional space which is used to separate the examples into categories (Cos Juez et al. 2010,

p. 1179). This technique can also be adapted to be used for regression, being named in this case support vector regression (SVR) (Bishop 2006, pp. 339-344). This regression capability would then allow predicting the throughput time in the considered example, an application already examined by several authors – e.g. Cos Juez et al. (2010, pp. 1177-1184) and Zhu et al. (2021, pp. 1-16). The throughput time consists, in this case, of the addition of processing and setup time (which can be predicted individually), disregarding the inter-operation time.

A characteristic of SVM is that they allow modelling nonlinear class boundaries (Witten et al. 2017, p. 252). This is done by utilizing kernel functions which map the data into a high dimensional feature space (Cos Juez et al. 2010, p. 1179). Several kernel functions are possible, this example will consider two: the polynomial kernel and the radial basis function (RBF) kernel. Depending on the kernel applied, the accuracy and performance can differ (Savas et al. 2019, pp. 1-16; Nti et al. 2021, pp. 3404-3411). Furthermore, such techniques are known to be sensitive to the parameters, requiring a careful selection of the corresponding values (Duan et al. 2003, pp. 41-59; Zhu et al. 2021, p. 2).

Several parameters are possible when using SVM (also depending on the programming environment utilized), worth mentioning are (Duan et al. 2003, p. 42; Bishop 2006, pp. 340-341; scikit-learn developers 2022):

- The regularization parameter  $C$ , which “determines the trade-off between minimizing the training error and minimizing model complexity”.
- The *degree* of the polynomial kernel function.
- The parameter *gamma* of the RBF kernel, which “defines how much influence a single training example has”.
- The parameter *epsilon*, which influences the epsilon-tube related to the error function in SVR.

Furthermore, the utilization of a preprocessing step is proposed in this example. The approach consists of utilizing clustering (in this case, the classical k-means technique) before performing the regression with SVR. This approach – already considered by other authors, for example Evgeniou et al. (2002, pp. 346-354) – proposes using the weighted

examples resulting from the clustering process to enhance the performance of the subsequent SVM.

Further data preparation steps, such as eliminating missing values, are not depicted in this example. This includes the possibly necessary conversion of nominal attributes in order to use the SVR technique.

The resulting process is illustrated in Figure 3-6, with the corresponding data mining phases. After training the SVR model, this needs to be tested, which results in a score measuring its accuracy. If acceptable, the model can be applied to predicting throughput times based on the considered attributes.

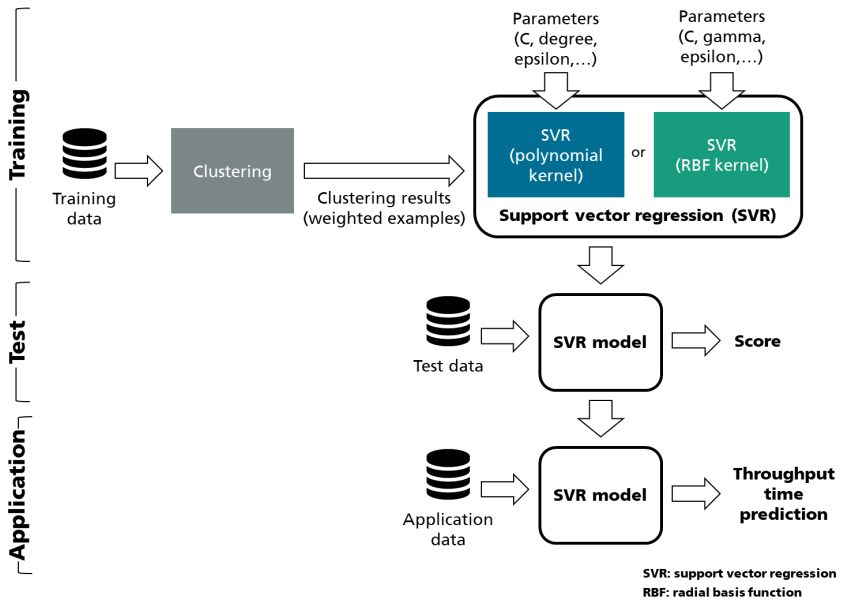


Figure 3-6: Data analytics example for predicting lead time

Two important characteristics of this example need to be highlighted:

- The combination of analytical techniques (clustering and SVR) create an analytical process.
- The performance of the SVR technique is highly dependent on the kernel and values of the parameters utilized. It is therefore necessary to provide a possibility to compare different configurations of the SVR technique in order to choose to most suitable one.

### 3.3.3 Summary

Innovative data analytics approaches offer the possibility to solve complex situations. Nevertheless, several aspects must be considered for their utilization: the different types of applicable analytics, their combination, the existence of a process for knowledge discovery with many iterations (between and within the steps), and the challenges during their applications. The objective is the creation of intelligent processes following the ideal of smart data.

A production management software can benefit from the advantages of data analytics. For this, it must take into account their capabilities and requirements. The features that a possible data analytics-based solution should present are shown in Table 3-1. The rows represent the requirements on a software-based production management system derived in chapter 2. The columns summarize the features of data analytics reviewed in this chapter that are relevant not only to assist production management but also to fulfill their own requirements in order to construct effective data analytics solutions. An "X" in the table represents a feature being able to address a requirement.

The considered features (with the corresponding section in parenthesis) are:

- the possibility to utilize different data analytics techniques (3.1.1, 3.1.2)
- the dynamic data-driven generation of specific and accurate analytical models (3.1.1)
- the flexible construction of flows of analytical algorithms/models (3.1.1, 3.1.3)
- the transfer of domain knowledge through pre-trained models (3.2.2)
- the transfer of domain knowledge through pre-built flows (3.2.2)

- the ability to select the most adequate available solution, considering different levels of detail and response (3.2.1)
- the ability to integrate with existing production management systems to enable data analytics, continuous learning, and create intelligent processes (3.2.3)
- the possibility to use data analytics synchronously and asynchronously (3.2.1)

The possibility to utilize different data analytics techniques refers to the usage of both, model-based and data-driven approaches, and of four data analytics types. The dynamic generation of models strives for the permanent adaptation to changes in the manufacturing environment. The aim is to materialize continuous learning.

Important is the possibility to build flows of data analytics. This will allow for the combination of data analytics algorithms/models, the implementation of the steps of the KDD process, and the required iterations. As with individual models, it is possible to transfer the knowledge contained in pre-built flows.

There must be a form of integration with the manufacturing environment. This will not only provide the necessary data. It will also allow for data analytics techniques that require interaction with the environment and for continuous learning processes. This also includes methods for the timely adaptation (triggering of new learning) of models – for example, through performance evaluations based on the consequences of the output generated in the real system. Furthermore, the integration must provide the means for the output to be utilized in the real system. The objective is, as stated before, the creation of intelligent processes.

The analytical solutions employed must match the required detail and response level (reaction time) in the best possible way. The possibility to run data analytics synchronously (online mode) and asynchronously (batch mode) refers to the different moments of execution, resources needed, and running times in the lifecycles of data analytics. This allows the separate performance of the data mining phases (e.g. training and testing in batch mode, application in online mode), the separate execution of the KDD steps, and for eager and lazy learning.

The IT efficiency of the solution can be enhanced by the characteristics of data analytics: dynamic generation of models (instead of programming), knowledge transfer, selection of the most suitable solution, etc. Nevertheless, this requires an implementation manner that covers the necessary integration, flexibility, and KDD process view. If not provided, the costs of generating, adapting, and using data analytics may outweigh the gains.







# 4 Review of Relevant IT Architectures

In the IT context, architecture can be defined as “the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution” (IEEE 1471-2000, p. 3).

This chapter will introduce the concepts and elements of architectures relevant for the utilization of data analytics and the design of the intended solution.

In the first part, current architectural approaches for the utilization of data analytics are presented and explained.

In the second part, the approaches for the creation of software solutions – including data analytics software – are covered. A special emphasis is put on the aspect of modularization and the new opportunities emerging from cloud-based technologies.

The combination of elements from both areas will serve as the basis for creating a solution that allows the utilization of data analytics in production environments for Mass Personalization.

## 4.1 Architectures for Data Analytics

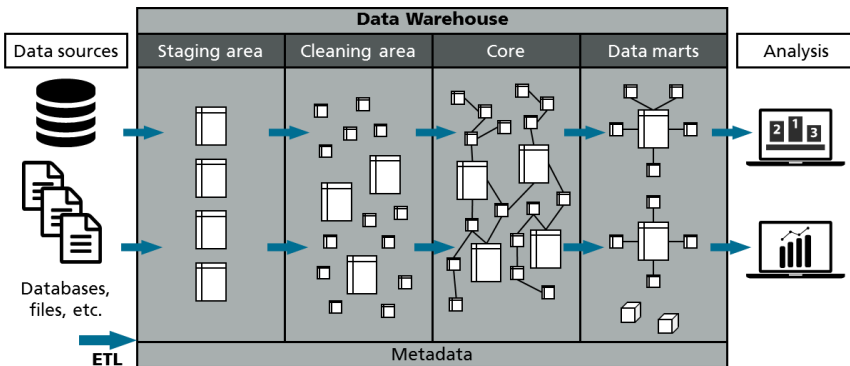
This section will deal with explaining current architectural approaches for the utilization of data analytics.

### 4.1.1 Data Warehouse

In order to assist during complex decision-making processes, **decision support systems** (DSS) were created. These are “interactive, computer-based systems intended to provide support to the decision makers engaged in solving various semi- to ill-structured problems

involving multiple attributes, objectives and goals” (Nemati et al. 2002, p. 144). Their task comprises the extraction of valuable information through data processing – performed using data analytics – for it to be used at the management level of enterprises.

DSS face a fundamental problem: they normally rely on heterogeneous operational data distributed among different non-integrated systems (Inmon 2005, p. 13). For this purpose, **data warehouses** (DW) were created. They consist of infrastructures that enable the integration and utilization of the application data found in the operational environment by providing the means to extract, clean, summarize, and store them (serving at the same time as a supply of historical data) (Nemati et al. 2002, p. 144; Inmon 2005, pp. 1-50). The term *infrastructure* refers to the hardware-intensive requirements to run this type of software solutions. The basic architecture of a data warehouse solution is depicted in Figure 4-1.



**Figure 4-1: Basic architecture of a data warehouse**  
(Schneider et al. 2016, p. 6)

The main architectural components are:

- **Data sources** are the origin from where the operational data (distributed among several systems) is extracted. They are also referred to as **operational level** (Inmon 2005, p. 16)

- 
- The **staging area** is where the data from the different systems (data sources) are stored at first. The data are saved as they come, conserving their original structure (Schnider et al. 2016, p. 7).
  - The **cleaning area** is where the required transformations are made in order to integrate, clean, and prepare the data from different sources before loading them into the core of the data warehouse (Schnider et al. 2016, p. 7). Some authors consider the function of the cleaning area as within the staging area (Kimball 2009, p. 16). Others refer to this combined area where the data transformations take place as **work area** (Bauer et al. 2013, p. 55).
  - The **core** is the area where the integrated and prepared data are centrally stored. It serves as a source of combined and easy to find historical data (Schnider et al. 2016, p. 7). The combination of these three areas can be referred to as **atomic level**, combining integration with a certain level of detail (lower than the operational level) and summary (Inmon 2005, p. 16).
  - The **data marts** contain partial extracts specially prepared for the intended users of each mart. Each data mart is created to be used by a specific group (i.e. sales, controlling, etc.), containing only the relevant data in the required form (Schnider et al. 2016, p. 8). They are also referred to as **departmental level**, with a decreased level of detail, high summarization, and all pertinent measures for a rapid response (Inmon 2005, pp. 16-18).
  - The **analysis** area utilizes the data from the marts. Though it is common to think of data warehouses as enablers for BI solutions, other types of data analytics can be employed (Inmon 2005, p. 239; Kimball et al. 2013, p. 23). They are also referred to as the **individual level** (ad hoc and temporary) (Inmon 2005, p. 16).
  - The **metadata** describes the available data from several points of view: *technical* (data structures, format, etc.), *functional/business* (what the data is, where it comes from, their relationships, etc.), and *process* (logs of results of DW operations) (Kimball 2009, p. 116; Schnider et al. 2016, p. 8). It is therefore essential for the tasks of the data warehouse and the performance of the intended analysis (especially during exploration) (Inmon 2005, pp. 102-103).

Kimball (2009, 121–130) considers the architecture of data warehouses as consisting of two types of components: **data stores**, depicting the “the temporary or permanent landing places for data”; and **services**, referring to the operative functions.

The process (and set of functions) from the extraction of the operational data to the loading in the core of the data warehouse receives the name of **ETL** (extract, transformation, and load). As its name indicates, it consists of three steps:

- The **extraction**, which consists of copying the source data into the work area. Several strategies are possible: (1) periodical extraction, depending on the periods of the type of data; (2) extraction on demand; (3) event-driven extraction (e.g. an established number of changes is reached); and (4) immediate extraction (because of changes) (Kimball 2009, pp. 425-469; Bauer et al. 2013, p. 56).
- The **transformation** tasks, which can be divided into two categories (Bauer et al. 2013, p. 57). Those for data migration (also known as data integration) consist of: adaptation of data types, conversion of codings, standardization of strings, standardization of dates, conversion of units of measure, and combination and separation of attribute values (Kimball 2009, pp. 439-463). The others deal with the more complex area of data cleaning.
- **Load**, which is “the physical structuring and loading of data into the [...] target dimensional models” (Kimball et al. 2013, p. 20). This step is required for transferring data to the core and the data marts. It must cover both types of data for the intended analysis, the specific (e.g. aggregates) and the independent one (Bauer et al. 2013, p. 58).

The **data warehouse manager** is responsible for the centralized administration of all components in the data warehouse, including the ETL process.

A particularly relevant component type used by the manager are the **monitors**, which detect relevant changes in the data sources (Bauer et al. 2013, p. 43). These can use different strategies: (1) log-based, checking the log for changes; (2) trigger-based, associating a trigger to a particular change; (3) replication-based, replicating the changes in other tables; (4) timestamp-based, using timestamps to detect changes; and (5)

snapshot-based, comparing snapshots to detect changes (Vavouras et al. 1999, pp. 87-89; Bauer et al. 2013, p. 54).

The data warehouse manager also administrates the components responsible for assisting the analytical functionalities – for example, by providing the necessary data in the required form (via the ETL process) and storing results from analysis in the data marts (Bauer et al. 2013, pp. 43-45).

The term **OLAP** (online analytical processing) denotes the utilization of special software techniques to enable fast, interactive, and diverse accesses to the data stored in the core and data marts (Gabriel et al. 2011, p. 52). They are based on the utilization of dimensional models to organize the way data are stored (Kimball et al. 2013, pp. 8-9). They can use either relational databases (ROLAP), multidimensional databases (MOLAP) – where data are stored as *OLAP cubes* – or a hybrid combination of both (HOLAP) (Bauer et al. 2013, p. 241).

The open character of the data warehouse architecture allows diverse forms of access to its integrated databases (Dittmar 2004, pp. 373-374).

#### 4.1.2 Extended Requirements

The increasing requirements on data analytics systems – described in chapter 3 under Big Data – cause for changes in the utilized architectures.

Marz et al. (2015, pp. 7-9) describes eight desired properties of Big Data systems:

1. **Robustness and fault tolerance** allow systems to deal with the complexity of distributed systems and the emerging errors (usually human).
2. **Low latency reads and updates** ensures that the applications are provided with current data as required.
3. **Ad hoc queries** supports the knowledge discovery process by allowing mining a dataset arbitrarily.

4. **Scalability** is “the ability to maintain performance in the face of increasing data or load by adding resources to the system”.
5. **Generalization** allows supporting and being utilized by a wide variety of applications.
6. **Extensibility** “allow[s] functionality to be added with a minimal development cost”.
7. **Minimal maintenance** allows the reduction of running costs of a system – for example, by using components with as little implementation complexity as possible.
8. **Debuggability** allows finding the causes of errors.

Several properties (in particular the first three) refer to the way data are utilized. This is addressed in the Big Data context with the **Lambda Architecture**, which consists of three layers, each one building upon the functionality provided by the layers beneath it (Marz et al. 2015, pp. 14-20). These are, from bottom to top:

1. The **batch layer** allows preprocessing queries on all the data available at a specific moment in time. It must be able to “store an immutable, constantly growing master dataset, and compute arbitrary functions on that dataset”. The precomputed queries are stored as batch views.
2. The **servicing layer** “is a specialized distributed database that loads in a batch view and makes it possible to do random reads on it”. It automatically replaces the old batch views with the new ones when they become available.
3. The **speed layer** produces views based on recent data, thus allowing the representation of the data that could not be taken into account during the batch precomputation (as they were generated after it started). These real-time views are constantly updated with the new data available (incremental approach). When the data are stored in the batch layer the corresponding results are deleted from the real-time views.

The first two layers cover almost all of the desired properties. The speed layer allows for low latency updates (a property missing in the other two).

In recent years, a new architecture named **Kappa** has emerged in response to the complexity of the Lambda Architecture. This proposes providing the same services as the Lambda Architecture but simplifying it by considering only data streams. In this view,

batch processing “is simply streaming through historic data”. The trade-off is, however, the reduction in speed and efficiency (increased latency) when compared to a true batch processing engine (Lin 2017, pp. 62-64).

Data analytics must also be able to use scalability, causing the utilization of parallel, distributed, and incremental mining algorithms (Han et al. 2012, p. 31). Similar to synchronous and asynchronous execution, this also requires a corresponding enabling architecture (current approaches will be covered in chapter 5).

To sum up, architectural approaches for data analytics are necessary in order to integrate and utilize heterogeneous data distributed among different systems. Data warehouses fulfil this objective by using different areas in order to read, cleanse, merge, prepare, and store the data – thus covering the steps of the ETL process – to finally allow for their analysis. The summarization and structuring of data in marts enable a rapid and simple access to the data. This, however, requires determining first which data is necessary and using adequate monitors to keep it up to date. Big Data increases the requirements on this approach for aspects such as robustness, scalability, and latency. The corresponding solutions (e.g. the Lambda Architecture) are based on layers that allow for previously preprocessing data (batch and serving layer) while also providing real-time views (speed layer).

## 4.2 Software Architectures

This section will describe the relevant architectural approaches used for building software solutions. Furthermore, the aspects of modularization and its subsequent influence on the software architecture are explained.

A summary comparing the different approaches is presented in section 4.3.

### 4.2.1 Monolithic Approach

Several definitions can be considered for the monolithic architecture depending on the point of view. Wolff (2018, p. 3) refers to a **monolith** as a “large software system that can only be deployed as a whole at once” having to “pass as a whole through all phases of the continuous delivery pipeline such as deployment, testing, acceptance and release”. Dragoni et al. (2017, p. 196) defines it as “a software application whose modules cannot be executed independently”. Dowalil (2018, p. 102) considers a monolith as a synonym for a “non-distributed system of uniform technology”.

Some monoliths are called Big Ball of Mud which refers to a system with “multiple tangled models without explicit boundaries” (Vernon 2016, p. 17). Although many monolithic systems fulfil this definition, this does not necessarily have to be the case (Brown 2014). The term can apply to both monolithic and distributed systems, depending on how they are developed (Dowalil 2018, p. 102).

Despite their many critics, monolithic systems possess a series of advantages originating in their simplicity. Some are (Dowalil 2018, p. 102):

- Debugging and troubleshooting are relatively easy.
- Batches, reports, and similar (predefined) complex procedures can be executed without major problems.
- Interfaces can be easily changed with no impact on other parts of the system (these are easy to detect in the code).
- The diversity of the technologies used is known and limited.
- Problems with communication over the network can be avoided.

Despite being usable in several fields, many modern applications discover problems in the monolithic architecture. Some of these are:

- Large-size monoliths become too complex, making them confusing, difficult to maintain, and tedious to debug (Daya 2015, p. 6; Dragoni et al. 2017, p. 196).



- “Monoliths [...] suffer from the *dependency hell*, in which adding or updating libraries results in inconsistent systems that do not compile/run or, worse, misbehave” (Dragoni et al. 2017, p. 196).
- The scalability of monoliths is limited (Dragoni et al. 2017, p. 196). Using a monolithic system “requires [...] to scale the entire application even though bottlenecks are localized” (Daya 2015, p. 6).
- “Monoliths [...] represent a technology lock-in for developers, which are bound to use the same language and frameworks of the original application” (Dragoni et al. 2017, p. 196).

Monolithic systems sacrifice flexibility in order to retain a simple construction (as long as size allows) and be robust; this last characteristic is however limited, as these systems are hard to maintain.

#### 4.2.2 Modularization

The concept of software modularization was introduced by Parnas (1972b) – based partially on the ideas of Gauthier et al. (1970) – and refers to the way a system is decomposed into modules. Each **module** represents an independent part of the system responsible for the execution of a specific task (or related group of tasks) with well-defined interfaces.

A related concept is that of **modularity**. According to Booch (2007, p. 56), it is “the property of a system that has been decomposed into a set of cohesive and loosely coupled modules”. This definition introduces two important concepts:

- **Coupling** is defined by Stevens et al. (1974, p. 117) as “the measure of the strength of association established by a connection from one module to another. Strong coupling complicates a system since a module is harder to understand, change, or correct by itself if it is highly interrelated with other modules”. Strong coupling leads to the utilization of obscure and complex interfaces. On the other hand, loosely coupling means “minimizing the dependencies among modules” (Booch 2007, p. 56).

- **Cohesion** “measures the degree of connectivity among the elements of a single module” (Booch 2007, p. 113). Yourdon et al. (1979, p. 106) expresses that “clearly, cohesion and coupling are interrelated. The greater the cohesion of individual modules in the system, the lower the coupling between modules will be”.

In order to be usable, modularity needs to be complemented by the two following concepts:

- **Abstraction**, which “denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer” (Booch 2007, p. 44).
- **Encapsulation**, which refers to the implementation of measures to achieve information hiding. The intention of the information hiding principle, introduced by Parnas (1972a, p. 342), is that all characteristics that are likely to change should remain hidden in order to facilitate internal modifications.

All three concepts – modularity, abstraction, and encapsulation – are required. Modularity covers the design decisions, trying to create systems with cohesive and loosely coupled modules. Abstraction refers to the observable behavior of each module or, in other words, to their interfaces. Encapsulation allows the implementation of the internal elements of each module and, at the same time, of the explicit barrier of the modules (thus implementing the abstraction) (Booch 2007, p. 51; Clyde et al. 2017, p. 109; Goll 2018, pp. 5-6).

Based on the covered concepts (and, by extension, on the works of authors such as Parnas, Gauthier, and Ponto) Goll (2018, pp. 8-9) recognizes the following characteristics in the modules integrating a system:

- In its design, each module follows the single responsibility principle – stating that “a module should be responsible to one, and only one actor” (Martin 2018, p. 62) – which guides that it should contain the elements that change together (cohesion).
- Each module can be used and tested without knowing its internal structure (abstraction).

- Each module can, thanks to its independence from other modules (loose coupling, abstraction, and encapsulation), be separately developed, tested, changed, and maintained.
- Each module can be developed and utilized independently from its context (how the system is finally implemented), being only dependent on the context communicated via its interface (loose coupling and abstraction).
- Each module can potentially be reused with little effort (cohesion, loose coupling, abstraction, and encapsulation).

At the same time, Goll (2018, p. 8) mentions the resulting following advantages of modularization:

- Complex systems become more manageable as a result of the low dependence between the modules.
- Systems become more flexible, as changes affect only a few modules (ideally one) and not the whole system.

Modularization can be employed to partially address the flexibility issue of monolithic system (Brown 2014; Dowalil 2018, p. 103). However, the resulting modular monoliths conserve many of the disadvantages of pure monoliths. Some worth mentioning, apart from the size of the modules, are:

- “Any change in one module of a monolith requires rebooting the whole application”, generating considerable downtimes (Dragoni et al. 2017, p. 196).
- “All modules must be brought together into production” (Wolff 2018, p. 3).
- “When choosing a deployment environment, the developer must compromise with a one-size-fits-all configuration, which is either expensive or sub-optimal with respect to the individual modules” (Dragoni et al. 2017, p. 196).
- As the modules share resources such as memory, databases, and files of the same machine, they are not independently executable (Mazzara et al. 2020, p. 31).

Object-oriented programming (OOP) and aspect-oriented programming (AOP) can be utilized in order to improve the modularity of the classical programming paradigms: structured and functional programming (Kiczales et al. 1997, pp. 220-240; Eden et al.

2006, pp. 113-124; Booch 2007, p. 43; Przybylek 2010, p. 139). This is further pursued by the utilization of services and microservices.

### 4.2.3 Distributed Systems

Evolution in computer technology, mainly the development of microprocessors and high-speed networks, has made it feasible and easy to construct computing systems composed of networked computers, be they large or small. Van Steen et al. (2017, pp. 1-5) defines the emerging **distributed systems** as “a collection of autonomous computing elements that appear to its users as a single coherent system”. A computing element – normally referred to as a node – can be either a hardware device or a software process.

According to the definition, distributed systems are based on independently acting elements – a principle of distributed systems – albeit in communication with each other, distributed among networked computers. The single coherent view intends for the users to notice this distribution as little as possible – this is known as *distribution transparency*. They often possess a software layer known as *middleware*, which allows for the communication between the distributed components, while hiding the differences in hardware and operating systems.

The utilization of such systems is not without problems. Rotem-Gal-Oz (2008) explains the fallacies formulated by Peter Deutsch to describe the issues faced in distributed systems. They are named like this because they originate from underestimating challenges such as network reliability, bandwidth, latency, or security.

Trade-offs are the advantages obtained by utilizing such systems:

- The execution environment for each software component can be flexibly chosen, selecting the one that is able to provide the performance that meets the requirements (Dowalil 2018, p. 101).
- Distributed systems are created with the intention of sharing resources (e.g. storage, peripherals, etc.), allowing also for a reduction of costs (van Steen et al. 2017, p. 7).

- An additional effect on the performance of distributed systems originates in the fact that they are built to be scalable (van Steen et al. 2017, p. 15).
- As long as they are able to communicate with each other, the technologies of each software component can be chosen individually (Dowalil 2018, p. 102).
- The distribution and network characteristics of distributed systems – especially since the introduction of mobile and embedded computing devices – allow them to be pervasive (naturally integrating into the environment) and continuously present. This results in the constitution of *ubiquitous systems* (van Steen et al. 2017, pp. 40–41). Further important features of such systems are, according to Poslad (2009, pp. 13-17): (1) context-awareness, optimizing the work in accordance to the context; (2) autonomy, being “self governing and [...] capable of their own independent decisions and actions”; and (3) intelligence, which, complementing the other two (similar as in section 3.2.3), “can enable systems to act more proactively and dynamically”.

The term **scalable** refers to the capability of a system to “handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity” (Neuman 1994, p. 463). Van Steen et al. (2017, pp. 20-23), based on the work of Neuman (1994), describes the techniques used in distributed systems for *scaling out* (expanding the system with more machines):

- *Hiding communication latencies* means utilizing, if possible, asynchronous communication between the applications (to avoid waiting for a response). This is often used in batch-systems and parallel applications, where the execution of tasks is scheduled while another task is waiting.
- *Partitioning and distribution* “involves taking a component, splitting it into smaller parts, and subsequently spreading those parts among the system”.
- *Replication and caching* refers to making copies of a component and distributing the created instances correctly among the system.

The simple improvement of hardware capacity is referred to as *scaling up*.

In recent years, the characterization of distributed systems was greatly affected by the introduction of **cloud computing** (van Steen et al. 2017, p. 30). The most widespread

definition may be that of Mell et al. (2011, p. 2) stating that “cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. He also describes its essential characteristics:

- on-demand self-service, allowing for the automatic provision of computing capabilities when required by the consumer
- broad network access, considering the communication with heterogeneous devices
- resource pooling, assigning and reassigning “different physical and virtual resources dynamically [...] according to consumer demand”
- rapid elasticity, allowing capabilities to be “elastically provisioned and released” (sometimes automatically) in order to scale according to demand
- measured service, meaning that the resource use is controlled and optimized automatically by leveraging a metering capability

These characteristics are provided through the *cloud infrastructure*. It consists of a physical layer (the hardware resources) and an abstraction layer (the software manifesting the characteristics) sitting above the former (Mell et al. 2011, p. 2)

Additionally, Mell et al. (2011, pp. 2-3) describes the three (widely accepted) basic service models in cloud computing:

- **Software as a Service (SaaS)** enables the consumer to utilize applications of a provider running on a cloud infrastructure. The consumer manages neither the application capabilities nor the underlying cloud infrastructure.
- **Platform as a Service (PaaS)** enables the consumer to deploy consumer-created or acquired applications onto the cloud infrastructure. These are based on programming resources supported by the provider. The consumer does not manage the underlying cloud infrastructure but is able to control the deployed applications and the setting of the execution environment to some extent.
- **Infrastructure as a Service (IaaS)** enables the consumer to run any desired software applications and operating systems using computing resources of the provider. The

consumer does not manage the cloud infrastructure but is able to control its own software, the storage, and the networking components.

Cloud computing allows reducing investments and operations costs related to the required hardware, replacing them with pay-per-use models. However, the evaluation of the gained advantage is, in many cases, not as simple and straightforward. Enterprises should weigh the benefits and costs involved before deciding on migrating applications to the cloud (Hajjat et al. 2010, pp. 243-254).

Given the requirements of some applications for characteristics such as low latency, quick processing of real-time data, or even the secure handling of sensitive data; approaches extending the cloud nearer to the field and devices (particularly in an IoT context) emerged (Kaur et al. 2020, p. 63). Fog computing is defined by the OpenFog Consortium (2017, p. 22) as “a horizontal system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum”. A step further is edge computing which, as name the states, locates processing at the edge of the network and very close to the devices (without necessarily being in the devices) (Yousefpour et al. 2019a, p. 294). This design allows for the support of data analytics enabling, for example, the utilization of deep learning by performing preprocessing tasks on the fog or edge level (Huang et al. 2017, p. 1; Kaur et al. 2020, p. 68), providing also advantages such as the enhancement of sensor fusion and the reduction of bandwidth and latency (Yousefpour et al. 2019b, p. 25).

#### 4.2.4 Service-oriented Computing

The next step towards the improvement of modularized systems is **service-oriented computing**, which finds its origins in object-oriented programming (Dragoni et al. 2017, p. 200). This, as the name indicates, is based on the utilization of services which, from a general perspective, are a “software program that makes its functionality available via a published API [application programming interface] that is part of a service contract” (Erl et al. 2017, p. 21). A service can be considered a collection of capabilities made available

through its interface (Erl et al. 2017, p. 23). The Open Group (2009, pp. 1-2) defines further characteristics, by considering that a service is

- a logical representation of a repeatable business activity that has a specified outcome,
- self-contained,
- capable of being composed of other services, and
- a *black box* to consumers of the service.

Service-orientation is based on a distributed solution logic which finds its roots in a theory known as **separation of concerns**. This states that “a large problem is more effectively solved when decomposed into a set of smaller problems or *concerns*”. The service-orientation approach addresses this concept with the utilization of logical units (services), each solving a particular concern while remaining agnostic to the main problem. The agnostic character refers to the logic being generic – not specific to a particular task – enough to allow its reuse in other business processes (Erl et al. 2017, pp. 23-24).

Thus, while distributed systems pursue a performance improvement through division of software components (technical view), service-orientation does it in order to improve the solution logic (functional view).

Booch (2007, p. 52) considers that encapsulation leads to a clear separation of concerns by providing “explicit barriers among different abstractions”.

Another important concept for the design of service-oriented solutions is **Conway’s law**. This states that “organizations which design systems [not only software] are constrained to produce designs which are copies of the communication structures of these organizations” (Conway 1968, p. 31). The law – although it is actually an observation – is relevant for the development of service-oriented structures in two ways. On the one side, it expresses that if the modularity of a solution is congruent with the structure of the organization (formal and informal) it will increase its efficiency. Such solutions are more usable (as they fit the way the processes are shaped), accepted, comprehensible, and transparent; allowing at the same time for an easier implementation and the reduction of changes (as the solution does not required to be adapted) (Dowalil 2018, pp. 7-8). From



the development side, it allows working in independent teams, each concentrating on one part of the solution (Wolff 2018, pp. 41-42).

Conway's law is also referred to as the mirroring hypothesis (Colfer et al. 2016, p. 711).

The last paragraphs explained how service-orientation works at a high level. On a low level, Erl et al. (2017, pp. 26-29) defines eight principles that should guide the design of services:

- **Standardized service contract:** The service contract communicates the capabilities of a service. The standardization allows for the communication (description of capabilities and data types) to be consistent.
- **Service loose coupling:** This reduced dependency “promotes the independent design and evolution of service logic while still guaranteeing baseline interoperability” – the latter being the ability to share information between two components and use it (IEEE 610.12-1990, p. 42). Services are designed to be natively interoperable (reinforced by the standardized service contract).
- **Service abstraction:** Only essential information about the service can be found in the service contract (the main source of information).
- **Service reusability:** The agnostic character of the service logic allows for it to be reused. The multipurpose logic in each service is encapsulated, facilitating the sharing and reutilization in different business processes (Erl et al. 2017, p. 373).
- **Service autonomy:** Services require a high level of control of their underlying execution environment in order to operate consistently and reliably.
- **Service statelessness:** “Services minimize resource consumption by deferring the management of state information when necessary” (for example, in a database). This also allows for a better scalability, as the service can be distributed on several nodes, where they become stateful (Wolff 2018, pp. 150-151).
- **Service discoverability:** Services need to be identifiable and understandable in order to be usable and reusable. This is achieved by utilizing adequate metadata.

- **Service composability:** In order to build sophisticated service-oriented solutions, services must be able to be part of compositions, regardless of their size and complexity.

#### 4.2.5 Service-oriented Architecture

The service-oriented architecture (SOA) introduces an architectural style based on the service-oriented design paradigm (The Open Group 2009, p. 1). This makes use of the technology of distributed systems in order to attain the design objectives of a service-oriented solution (Erl et al. 2017, p. 61).

Based on the service-oriented design principles and the advantages provided by the distributed systems technology, Erl et al. (2017, pp. 61-69) identifies four main characteristics that service-oriented architectures should have:

- They should be **business-driven**, meaning that the business vision, goals, and requirements are the guide for the architectural model. The service-oriented technology should be flexible enough to evolve with the organization, maximizing its value and lifespan.
- They should be **vendor-neutral**, not being dependent on a proprietary vendor platform. This allows not only for the utilization and combination of different technologies but also for the evolution of the system in order to continuously fulfil and adapt to the business requirements in the best way possible.
- They should be **enterprise-centric**, meaning that the services should be considered enterprise resources. As such, they do not “belong” to a specific application and can be freely used (consumed) by other software in the enterprise. This allows them to be part of larger software solutions. This characteristic is highly dependent on the reusability and standardized communication (interoperability) of the services.
- They should be **composition-centric**, referring to the ability of services not only to be reused but to behave as “flexible resources that can be plugged into different aggregate structures for a variety of service-oriented solutions”.

The communication between services is performed via messages (pieces of data) (Narkhede et al. 2017, pp. 45-48). A *Message Bus* is a simple pattern from the EAIP (Enterprise Application Integration Pattern) family, consisting of a transport mechanism for asynchronous messages between services guided by endpoints (emitting and consuming services). A *Message Broker* is a more complex pattern, having the responsibility of deciding to whom a message must be delivered (Dowalil 2018, pp. 76-77). It works based on a pattern known as publish/subscribe “characterized by the sender (publisher) of a piece of data [...] not specifically directing it to a receiver, instead, the publisher classifies the message somehow, and that receiver (subscriber) subscribes to receive certain classes of messages” (Narkhede et al. 2017, p. 1).

The communication between services can be synchronous or asynchronous. Additionally, two types of collaboration emerge: *request/response*, in which a service waits for an answer (normally used for synchronous communication, although it can be used for asynchronous); and *event-based*, in which services subscribe to the events to which they should react (used for asynchronous communication) (Newman 2015, pp. 42-43).

In accordance with the enterprise-centric and the composition-centric characteristics, services can be composed to constitute processes (in line, as per Conway’s law, with the business processes). Two architectural styles can be used to implement and manage the corresponding sequence of services (Newman 2015, p. 43):

- **orchestration**, which utilizes a central instance that “conducts” the process
- **choreography**, in which every component is aware of its tasks and must perform them autonomously

The advantage of orchestration is that it provides a good control of the (synchronous) process. Its disadvantages are that it creates dependencies between the services – in a point-to-point integration (D’Amore 2015) – and the total processing time is the sum of the processing times of each service. Furthermore, if the central instance stops working, the whole processing stops (Bonham 2017).

The choreography, on the other hand, is based on services reacting to events to which they subscribe. In consequence, the dependency between the services is very low,

allowing them to be added and removed as required. Furthermore, a faster end-to-end execution is possible, as the services can be executed asynchronously. Also, there is no central instance serving a single point of failure. The disadvantage is that the processes become unclear, being only modelled implicitly and with the complexity shifted from a central instance to the individual service logic (Bonham 2017). This requires more monitoring and tracing effort (Newman 2015, p. 45). This issue will be even worse in complex processes.

Although both styles can be applied, SOA solutions are regarded as frequently using orchestration (Wolff 2018, pp. 84-94). This is due to its simplicity and easier ways to manage complexity (Dragoni et al. 2017, p. 203).

An *integration platform* is where the communication between services takes place. It is also regarded as the place where the services are composed using orchestration. The communication with external consumers, on the other hand, is usually done by means of a *portal*. This offers an interface through which the services can be used (Wolff 2018, p. 85).

#### *Advantages and Disadvantages of SOA*

Based on the features of service-oriented design and the characteristics of SOA, solutions using a service-oriented architecture should present the following advantages:

- The implementation of the separation of concerns through services enables the division of complex problems in simpler services (Dragoni et al. 2017, p. 200).
- Services are easily reusable, allowing also for a reduction of application-specific logic and of the volume of logic overall (Erl et al. 2017, pp. 35-36).
- The composition of services can be flexibly changed, allowing for good maintainability and extensibility, and in turn increasing the ability of the enterprise to react agilely to changes (Erl et al. 2017, pp. 50-51). Furthermore, the composition of services into complex applications enables the generation of new utilizations of existing services (Takai 2017, p. 17).

- The consistent utilization of service-oriented systems allows reducing waste, redundancy, size, and operational costs of IT systems, thus increasing cost-effectiveness and IT efficiency (Erl et al. 2017, pp. 52-53). This factor is also reinforced by the reusability, maintainability, and extensibility of SOA systems;
- The reusability of services, the flexibility (maintainability and extensibility) of SOA systems, and the increase in IT cost-effectiveness improve the return on investment (ROI) of service-oriented solutions (Erl et al. 2017, pp. 48-49).
- The vendor-neutral characteristic enables the utilization of different technologies and approaches, improving the effectiveness of the solution and allowing it to evolve (Erl et al. 2017, pp. 63-65).
- SOA – using characteristics of distributed systems – provides a good scalability for services, by being able to choose the best execution environment for each service (Dowalil 2018, p. 101). The effect is enhanced by the division of processes into services (concurrency) – allowing to execute each part of the process in the most appropriate manner – and by splitting the load using several instances of the same service processed in parallel in different nodes (partitioning) (Dragoni et al. 2017, p. 200; Fowler 2017, p. 5).
- The integration of the IT landscape with other heterogeneous systems is improved, as it requires merely the implementation of standards protocols to communicate (Dragoni et al. 2017, p. 200).
- SOA allows for different development teams to work separately on their own applications (services) (Wolff 2018, p. 85).

Furthermore, the utilization of distributed systems allows using the benefits covered in section 4.2.3. This includes the usage of the service models of cloud computing – for example, IaaS to reduce infrastructure costs.

However, the mentioned benefits refer to those which are desired when designing service-oriented solutions. Traditional SOA systems present a number of issues product of how they are actually realized:

- Perhaps the biggest problem is the fact that the use of orchestration tends to generate a sturdy execution structure, hindering loose coupling by generating dependencies between the services and causing the de facto creation of monoliths, with all associated problems. In extreme cases, most of the logic is transferred to the central orchestration instance, with the services acting only as data managers (Wolff 2018, p. 88).
- These rigid compositions generated by orchestration also cause for a loss of flexibility, since making changes requires modifying program code and deploying the whole application, with its corresponding effort (Wolff 2018, p. 88).
- SOA systems are complex, with elevated requirements regarding performance and scalability, and with the network problems characteristic of distributed systems (Takai 2017, p. 16).
- As changes in a service can affect many users, modifications of the interface – the exposed part, thanks to the encapsulation – can become tedious in SOA systems. To overcome this problem, versioning becomes necessary, in order to manage old interfaces (Wolff 2018, p. 87).
- Debugging can be challenging due to the need to reproduce the state of orchestrated environments and because of the demand to deliver fast solutions in order to reduce downtime (Arora et al. 2018, p. 452) (an issue amplified by the characteristic of the central orchestration instance as single point of failure).
- The initial investment to migrate whole legacy systems into the SOA solution can be considerable (Wolff 2018, p. 86).

Mazzara et al. (2020, p. 32) goes one step further by affirming that “SOA has no focus on independent deployment units and related consequences, it is simply an approach for business-to-business intercommunication”.

#### 4.2.6 SOA Reference Architecture

There are many approaches to describe the constitution of SOA. This work will consider the SOA Reference Architecture (RA) described in the ISO/IEC 18384-2:2016, which was created together with The Open Group with the intention of normalizing the concepts involved. The proposed reference architecture is illustrated in Figure 4-2. The architectural building blocks (ABBs) – “logical elements that supports realization of one or more capabilities” – are organized into two types of areas (ISO/IEC 18384-2:2016, p. 10):

- Functional **layers** are “abstraction[s] of a grouping of a cohesive set of related capabilities and their identified ABBs, interactions among ABBs, interactions among layers, and the influences on and by architectural decisions”.
- **Aspects** are a type of layer that “contains capabilities and functionality that are widely useful across functional layers and may need to be coordinated across multiple roles”.

The aspects can be regarded as transversal to the horizontal functional layers.

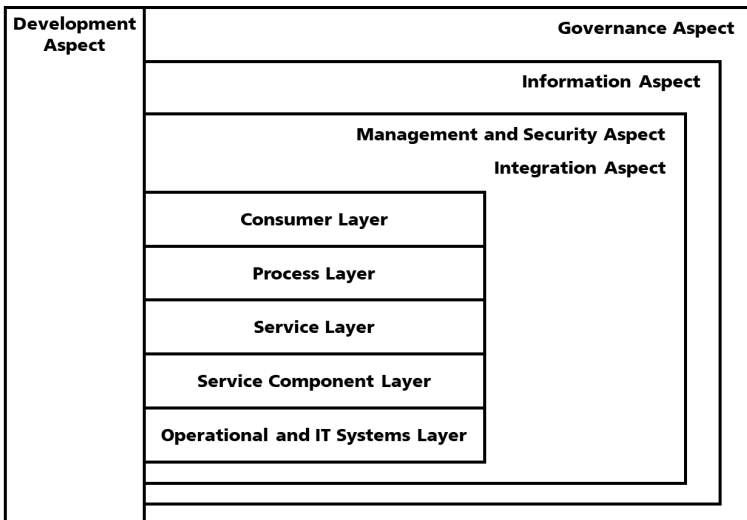


Figure 4-2: Reference architecture for SOA solutions (based on ISO/IEC 18384-2:2016)

The **Operational and IT Systems Layer** is the intersection between SOA and the organization infrastructure. It covers design, deploy, and runtime. Furthermore, it represents the existing software systems – for example, legacy applications (ISO/IEC 18384-2:2016, pp. 30-31).

The capabilities of this layer encompass:

- *service delivery*, managing the functional delivery of services, including their technical implementation and acting as a broker between them and the invoking applications (other systems)
- *runtime environment*, providing the runtime capabilities for the execution of services and other components of the RA
- *virtualization and infrastructure services*, managing the capabilities of the underlying infrastructure (computing power, storage, etc.) in a native or virtualized manner (ISO/IEC 18384-2:2016, p. 32).

The **Service Component Layer** contains, as its name indicates, service components. They support software components responsible for the functional implementation (or *realization*) and operation of services. Each service component can realize one or more services (ISO/IEC 18384-2:2016, p. 43).

The capabilities of this layer encompass:

- *service realization and implementation* (design time)
- *service publication and exposure* (design time)
- *service deployment* (design time)
- *service invocation* (runtime)
- *service binding* (runtime), which supports the service interoperability (ISO/IEC 18384-2:2016, pp. 44-45)

The **Service Layer** “contains the service descriptions for business capabilities, services and IT manifestation used and created during design time, as well as runtime service contracts and descriptions that are used at runtime” (ISO/IEC 18384-2:2016, p. 58).

The capabilities of this layer support the following responsibilities:



- *identification and definition of services*
- *provision of a container which houses the services*
- *enabling the use of a registry/repository that virtualizes runtime service access*
- *enabling the use of a registry/repository to house and maintain service design-time information (ISO/IEC 18384-2:2016, p. 59)*

The **Process Layer** “covers the process representation, composition methods, and building blocks for aggregating loosely coupled services as a sequence of steps aligned with business goals”. The emerging service processes – based on compositions of services – provide support for specific use cases and business processes (ISO/IEC 18384-2:2016, p. 69).

Relevant capabilities of this layer encompass:

- *process definition*
- *event handling*, managing the reaction to business events (e.g. creation of a document) which may need for a service process to be executed
- *process runtime enablement*, referring to the realization and deployment of service processes, and to the creation, management, and execution of their individual instances
- *process information management*, managing the information needs of a service process such as context and state information, performing the transformation of the data as needed within the process, and providing a process repository
- *process integration*, making the service process available to be used
- *process monitoring and management*, identifying bottlenecks and optimizing the workload in service processes (ISO/IEC 18384-2:2016, pp. 72-73)

Particularly relevant is the ABB known as *Process Engine*, which manages the execution of processes, their instances, and their context.

The **Consumer Layer** is “where consumers, either human actors or SOA solutions, interact with the SOA solution or ecosystem. It enables SOA solutions to support a client-independent, channel agnostic set of functionality, which is separately consumed and rendered through one or more channels (client platforms and devices)”. This

interaction takes place using interfaces, which may point to a service or process (ISO/IEC 18384-2:2016, p. 80).

Relevant capabilities of this layer encompass:

- *consumer services*, which enable the interaction with the service consumer, providing a response to a request
- *presentation services*, which support the presentation of information in a way that meets the requirements of the consumer
- *backend integration*, providing the integration with the underlying functional layers
- *information access*, sharing access to data and metadata through the Information Aspect (ISO/IEC 18384-2:2016, pp. 81-82)

The **Integration Aspect** is responsible for matching service requests and service implementations. It acts as a mediator between the service requester (the consumer) and the service provider, which are loosely coupled (ISO/IEC 18384-2:2016, p. 90).

The two main capabilities of this layer can be grouped into (ISO/IEC 18384-2:2016, pp. 91-95):

- *Communication, service interaction, and integration*, with the task of connecting the service requester with the service provider (including service discovery), even linking systems that do not directly support service-style interactions. A main ABB in this category is the *mediator*, which coordinates and handles the service request/response interaction.
- *Message processing*, with the tasks of performing “the necessary message transformation to connect the service requestor to the service provider and to publish and subscribe messages and events asynchronously”. An important ABB in this category is the *event broker*, which enables event consumers to subscribe to an event and event provider to publish the event.

The **Management and Security Aspect** manages the technical (non-functional) features and issues of the SOA system, ensuring that it “meets its requirements with respect to: monitoring, reliability, availability, manageability, transactionality, maintainability,

scalability, security, safety, lifecycle, auditing and logging, etc.” (ISO/IEC 13249-6:2006, p. 101)

The main capabilities of this layer include:

- *security management*, for example, administration of roles, access rights, system recovery, etc.
- *IT systems monitoring and management*
- *service and SOA solution monitoring and management*, with a focus on the technical performance
- *business activity monitoring and management*, elaborating metrics on the performance of business processes utilizing the services
- *configuration and change management*
- *policy management*, storing and implementing policies regarding security, data access, the business, etc. (ISO/IEC 18384-2:2016, pp. 104-108)

The **Information Aspect** is responsible for managing and providing data, metadata, and information (ISO/IEC 18384-2:2016, p. 121).

The main capabilities of this layer include:

- *information services*, which enable managing the available data
- *information integration*, which encompasses ETL tasks in order to provide the data in the required form
- *basic information management*, responsible for managing metadata and unstructured data
- *business analytics*, providing support for analytical functionalities
- *information definition and modeling*, creating a common information model
- *information repository*, which stores relevant information (ISO/IEC 18384-2:2016, pp. 122-124)

The RA contemplates business analytics only superficially. The considered functionalities act mainly as enablers for data analytics (e.g. providing information as basis for a BI system). No special advantage for data analytics is therefore derived from the serviced-oriented constitution.

The **Governance Aspect** “includes both SOA solution governance (governance of processes for policy definition and enforcement), as well as service governance (service lifecycle)”. The SOA Governance “defines policies, guidelines, standards and processes that reflect the objectives, strategies and regulations to which services and SOA solutions conform”, and which are in line with the business objectives (ISO/IEC 18384-2:2016, p. 136).

This aspect can then be considered as being constituted by two main capabilities (ISO/IEC 18384-2:2016, pp. 137-140):

- the *governance management*, providing the ability to plan, define, implement, enable, and monitor governance
- the *monitoring of SOA services and solutions* with regard to governance

The **Development Aspect** “contains all of the components and products needed to develop and change implementations of SOA services and solutions” (ISO/IEC 18384-2:2016, p. 151).

This aspect can then be considered as being constituted by the following main capabilities:

- *development*, with all the related tasks (e.g. debugging)
- *testing*
- *deployment*
- *publication*
- *maintenance*, fixing or extending services and solutions (ISO/IEC 18384-2:2016, pp. 157-159)

This section performs an overview of the layers and aspects in the SOA Reference Architecture. The ISO norm provides more details on responsibilities, capabilities, and internal dependencies (as the components of the layers communicate with each other).

Furthermore, throughout all the layers, elements of security and policy management are considered (and not only in the management and security aspect). However, these are not directly addressed within this work, as its focus lies on the functional view of the service-oriented solutions.

One important feature of the norm is that the reference architecture described is intended for creating a SOA system complementing other IT systems where the business processes are performed (e.g. an ERP system).

#### 4.2.7 Microservices

Building upon the service-oriented design and with the objective to address several problems of SOA – mainly its flexibility – a new development approach by the name of **microservices** was developed.

Newman (2015, p. 2) defines microservices simply as “small, autonomous services that work together”. Nadareishvili (2016, p. 6) provides a more technical concept by defining a microservice as an “independently deployable component of bounded scope that supports interoperability through message-based communication” while affirming that the **microservice architecture** “is a style of engineering highly automated, evolvable software systems made up of capability-aligned microservices”.

Microservices have the following characteristics, which also define the advantages of their utilization:

- Microservices are small and specialized in performing an individual functionality or business capability, in an approach following the single responsibility principle. This enables a very good modularity – with loosely coupled elements – and the utilization of bounded contexts. In this way, the small size is the reason for most of the following features and benefits (Newman 2015, pp. 2-3; Dragoni et al. 2017, p. 202; Wolff 2018, p. 32).
- Microservices are autonomous, each one being a separate and independent entity that can be changed and deployed by itself without requiring consumers to change (Newman 2015, p. 3). In contrast to SOA; where logic is normally distributed among individual services, the orchestration, and other tools (e.g. a portal); the isolation of a microservice allows it to contain the whole logic pertaining a functionality (Wolff 2018, p. 88).

- 
- The high modularity of microservices enables their composability, allowing the flexible modification of applications as required (Newman 2015, p. 7). This provides for good extensibility (Dragoni et al. 2017, p. 202).
  - In a similar way, the independency and individual deployment of small microservices allow for their replaceability. This provides for good maintainability, preventing the “erosion” of the architecture (Dragoni et al. 2017, p. 202; Wolff 2018, pp. 60-61).
  - The independency of microservices also allows for the use of a bulkhead pattern to increase resilience. As failures do not cascade, they can be isolated and the rest of the system can continue working (Newman 2015, p. 5; Dragoni et al. 2017, p. 204).
  - As long as the microservices are able to communicate with each other, the technology applied can be chosen freely. This allows selecting the right tools for the desired functionality – instead of one-size-fits-all approach. The architecture also allows testing new technologies with limited risks (Newman 2015, p. 4; Wolff 2018, p. 66).
  - Microservices – usually implemented as a distributed system – allow for high scalability, even better than in traditional SOA, due to the separation of processes in small and properly isolated components (Dragoni et al. 2017, p. 201; Wolff 2018, pp. 64-65).
  - As with SOA, microservices are easily integrable with existing systems of the IT landscape, requiring only the use of the right communication protocol. Thus, the functionality of a system can be extended without extensive modifications to its program code (Wolff 2018, p. 61). Dowalil (2018, pp. 135-136) refers to the way microservices can complement and extend monolithic systems.
  - Microservices constitute small and isolated units which are independently deployable. In this way, modifications to microservices can be brought rapidly into production (Newman 2015, p. 6). This is considered one of the sources of the flexibility of microservices (Wolff 2018, p. 94). The small size of the deployment units is a good complement for continuous delivery, which promotes bringing software into production in short and regular cycles, in a process which is ideally automatized. Managing small units, the process can deliver fast feedback. Besides, the risk is

minimized, as changes are small and isolated, being easy to undo (Wolff 2018, pp. 63-64).

- Small independent development teams can be organized around the microservices they are responsible for. This increases their productivity and allows for a better utilization of resources (Newman 2015, p. 7; Wolff 2018, pp. 67-69).

Microservices have a focus on functionality, emphasizing the support of business capabilities. Their maintainability and extensibility allows them to constantly evolve (Dragoni et al. 2017, p. 202), following the development of the business.

As microservices are thought to be continuously changing, they require to overcome the problems presented by orchestration in SOA solutions in order to enhance their flexibility (Wolff 2018, p. 89). For this reason, services are normally based on event-based choreographies, which allow to truly decouple the components (Newman 2015, pp. 43-46; Dragoni et al. 2017, pp. 203-204). All intelligence is concentrated in autonomously and asynchronously executable services and not some central coordinating instance. Each instance understands its role and communicates with the others (Wolff 2018, p. 89). This also contributes to increasing resilience, as microservices are decoupled and therefore a single point of failure, which is characteristic of orchestration, does not exist.

### *Elements and Integration*

The architecture of microservices is fairly simple and does not differ a lot from the one of standard applications. Each microservice consists of three main elements: (1) the *frontend*, constituted in this case by the API of the microservice which is in charge of handling the interactions; (2) the *backend*, containing the code to fulfil the functionality of the microservice; and (3) a *way to retrieve data*, which is either stored in memory or in a database (Fowler 2017, pp. 9-11). Each microservice has its own user interface, unlike SOA, where this is managed by a portal (Wolff 2018, p. 92).

A usual method for integrating services with each other and with applications is the utilization of a shared database (Newman 2015, p. 41). Though common, this method presents several challenges for its use with microservices:

- As the internal structure of the database is utilized by several applications, it cannot be easily changed; each change must be coordinated with other applications. This can also affect the speed by which changes to the related microservices can be implemented (Wolff 2018, pp. 187-188).
- Microservices which make use of the database are limited to the technology utilized by it (Newman 2015, p. 41).

However, measures can be applied to minimize these problems. For example, it is possible to separate the data sets used by each microservice, allowing them to use their own schema (Wolff 2018, p. 188).

An important alternative is the use of replication techniques. These allow copying the data into the desired schema. The trade-off is the generated inconsistency of the data, as it takes time until changes are replicated (Wolff 2018, pp. 188-189).

Another way of integration is the utilization of messages to share data. As with SOA, microservices utilize messages as their main way of communication. However, microservices emphasize the utilization of simple communication systems without own intelligence (Wolff 2018, p. 91). This is in contrast to SOA, where complex applications can be used to manage messages, such as, for example, an Enterprise Service Bus (ESB) (Dowalil 2018, pp. 77-78).

In the microservices context, standards are used for developing APIs. That can comprehend, for example, the technology they utilize for messaging or the way they are designed (Nadareishvili 2016, p. 32).

### *The Inverse Conway's Law*

The relation between the structure of an enterprise and the systems it employs was established in section 4.2.4 using Conway's law. Based on the restrictions established by it, ThoughtWorks (2015) proposes – following an article by Leroy et al. (2011) – the



utilization of the so-called Inverse Conway Maneuver, which consists of “evolving your team and organizational structure to promote your desired architecture”.

Bloomberg (2015) recommends further developing this concept by reversing the Inverse Conway Maneuver, affirming that the real discussion is “how companies can best leverage changing technology in order to transform their organizations”. Fowler (2017, pp. 21-22) refers to this idea as the **Inverse Conway’s Law**, stating that “the organizational structure of a company is determined by the architecture of its product”, being the “product” the system the organization produces and utilizes. She also affirms that, in consequence, the utilization of microservices will lead to an organization that is modularly structured with a high granularity. As the microservices they utilize, the components of the organization are autonomous and require building the necessary communication in order to cooperate and avoid problems.

#### *The Bounded Context*

Within Domain-Driven Design (DDD), Evans (2015, p. 2) recognizes the problems that can arise from combining contextual models into a larger application, causing for software to “become[s] buggy, unreliable, and difficult to understand”. This causes confusion in the communication among team members (developers). This issue is rooted in the fact that “It is often unclear in what context a model should not be applied”. Microservices, in their attempt to “break large components (models) into smaller ones in order to reduce the confusion and bring more clarity to each element of the system”, follow a style compatible with DDD (Nadareishvili 2016, p. 64).

To address the issues, Evans (2015, p. X) introduced the concept of *bounded context*, which is a “description of a boundary (typically a subsystem, or the work of a particular team) within which a particular model is defined and applicable”. Newman (2015, pp. 31-33) describes a domain (a business) as consisting of multiple bounded contexts where models “reside” which can either communicate with the outside of the bounded context (through its interface) or not. Following this contextual separation allows for the creation of microservices that are loosely coupled and strongly cohesive.

Because of their circumscription to a contextual boundary, microservices are considered as being non-agnostic. This is the cause for an important characteristic of microservices: their reusability is limited. Although the logic could be reused, this is restricted to the parent business process. Reusability, on the other hand, must be possible across business processes (Erl et al. 2017, pp. 113-114). Microservices are focused on business capabilities and not on code reutilization (Shadija et al. 2017, p. 2).

### *Challenges of Microservices*

In spite of all the described benefits of microservices, this approach also presents challenges that have to be considered:

- As with SOA, microservices architectures can become fairly complex and unclear. This is due to the great number of independently acting components (Daya 2015, p. 27) and is worsened by the utilization of choreographies, which only model implicit relationships. The complexity of the solution also represents a factor influenced by the size of microservices, as it increases with the size reduction (Newman 2015, p. 3).
- Also as with SOA, the debugging in microservices architectures is difficult (Dragoni et al. 2017, p. 210; Zhou et al. 2019, p. 1).
- Finally, like SOA, microservices architectures are – being a type of distributed systems – vulnerable to network problems, such as breakdowns and latency issues (Takai 2017, p. 22).

#### 4.2.8 Microservices in the SOA Approach

Although microservices and SOA are both based on service-oriented approaches, they differ in several aspects. Additionally to the already described dissimilarities, it should be taken into account that the objective behind the utilization of microservices is to “partition the components of a distributed application into independent entities, each addressing one of its concerns” (Dragoni et al. 2017, p. 197). Wolff (2018, pp. 90-94) considers microservices as being an architecture for individual projects, while SOA is an enterprise-

wide architecture. This allows for microservices to provide service-oriented solutions for single systems, which are light-weighted and cost-effective (by using less services).

However, many authors and researchers see beyond these differences, focusing on how microservices integrate with SOA in order to extend its functionality. Newman (2015, p. 9) refers to microservices as a technique emerging from the real world that can help implement SOA correctly, even considering it as a specific approach for SOA. In the same line of thought, Dowalil (2018, pp. 134-135) refers to typical SOA applications as SOA 1.0 and to the ones enhanced through the application of microservices as SOA 2.0. Erl et al. (2017, p. 113) views the SOA architecture as composed of a mix of agnostic (reusable) services and non-agnostic microservices.

### *Hybrid Approaches*

As SOA should profit from the flexibility gained through microservices – addressing its main disadvantage – it is advisable to think beyond the hard division in orchestration for SOA and choreography (reactive approach) for microservices. Bonham (2017) considers cases where combining both approaches makes sense, for example, with a solution composed of synchronous blocks (coordinated through orchestration) of asynchronous activities, or vice versa. To address such cases, he describes two hybrid approaches.

The **first hybrid approach** considers using choreographies between services, which in turn coordinate (orchestrate) subordinated microservices. The approach depicted in Figure 4-3 presents the benefits of using decoupled services which are asynchronously executable (through the events), each containing its own logic (distributing the processing flow). On the downside, the approach creates coupling between the orchestrated microservices.

Reasons for using this approach include:

- if most of the processing is (or can be) done asynchronously
- if speed to market is a priority
- if decentralizing the flow into each service is manageable
- if there are sequential steps that only apply within the orchestrating services

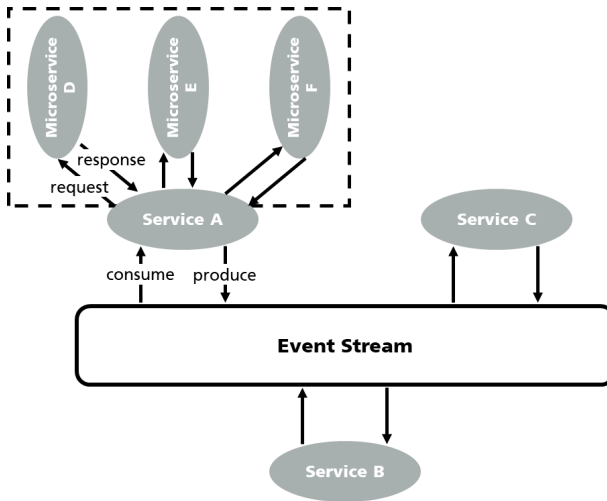


Figure 4-3: First hybrid approach (Bonham 2017)

This work will refer to the orchestrating entity as “service”, as it does not fulfil the characteristic of a microservice.

The **second hybrid approach** utilizes choreography between microservices and a central coordinator managing the flow. The latter acts similar to an orchestrator, producing events for what needs to be done (commands) and consuming events representing what has been done. This operation is depicted in Figure 4-4.

This approach provides the benefits of using decoupled microservices, enabling asynchronous processing (through the events), and being able to find the overall flow in the reactive coordinator. The trade-offs are, however, that a sort of coupling does exist between the coordinator and the microservices – as it needs to know the events for emitting commands and reacting – and the possibly serious impact of a failure in the coordinator.

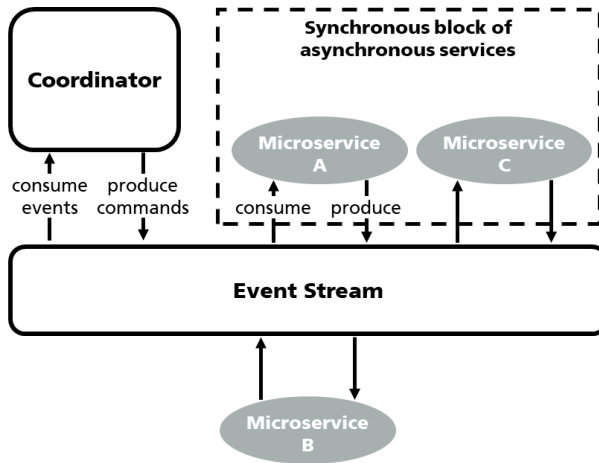


Figure 4-4: Second hybrid approach (Bonham 2017)

Reasons for using this approach include:

- if synchronous blocks of asynchronous processing are used (in the example, microservices A and C start at the same time)
- if the flow could change depending on the data being processed
- if it is desirable to see the end-to-end flow at design time and runtime
- if there is a need to decouple the microservices as much as possible

#### 4.2.9 Software Architectures in Throughput Time Prediction Example

Software solutions for implementing the analytical example for throughput time prediction presented in section 3.3.2 could be designed using the reviewed architectures. The particularities of each architectural approach applied to the example are illustrated in Figure 4-5.

The monolithic approach with some modularization elements (*modular monolith*) provides, as expected, the most simple and least dynamic configuration. In the blocks

representing the corresponding throughput time prediction modules (for either training and testing or applying the model), the analytical approaches are contained within procedures called following a hard-coded logic. In order to apply an SVR model with a specific kernel, the user must choose the corresponding application module. In the best-case scenario, this can be done by means of parameters in the main system utilizing the prediction module. In the worst-case scenario, it would require changes to the logic of the main system.

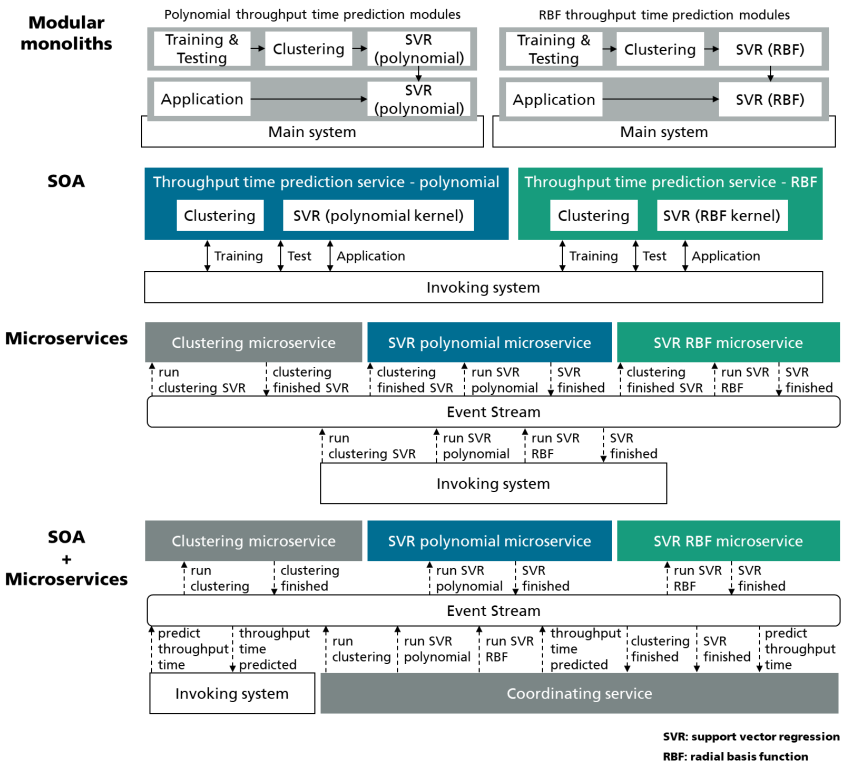


Figure 4-5: Architectural designs for throughput time prediction example

In the monolithic approach, changes to the model can only be done by specialists. This would require programming effort for the corresponding application module to use the new model. Furthermore, changes to the logic (e.g. adding a new analytical functionality) would require adapting the programming of all related modules.

The approach utilizing the service-oriented architecture (SOA) provides the analytical functionalities as services. The invoking system can decide which kernel to utilize by selecting the corresponding service when the process requires it. Furthermore, each service contains the possibility to train, test, or apply the SVR model depending on how it is invoked.

The approach decreases the amount of hard-coding, enhancing the software modularity (particularly when it comes to adding new services). However, it conserves problems of the monolithic approach by creating services with a great amount of internal logic.

The microservices approach achieves a great degree of flexibility by providing each microservice with the minimal functionality possible while remaining autonomous. The loose coupling between the microservices allows combining them in any manner required (or even adding more analytical functionalities without effort). Furthermore, the SVR microservices can be easily replicated and reused by other applications.

The trade-off of the approach is the resulting complexity. Pure microservices act autonomously according to events detected in the event stream. The SVR microservices, for example, need to react (subscribe) to events specifically calling them or indicating the end of the previous analytical step (in this case, clustering). Each event can also carry data (for example, indicating if the execution is for training, testing, or application purposes). The more microservices there are in a process, the less understandable it becomes.

The approach combining the SOA and microservices architectures attempts to reduce the complexity of the latter. A coordinating service containing the steps required for each execution type (training, testing, or application) manages the events in the correct order. Each process is clearly mapped within the coordinating service. This also allows reducing the number of events and providing a simpler interface to the invoking system.

It is worth mentioning that Colangelo et al. (2018, pp. 191-196) proposed a services-based concept similar to the ones depicted in the same area of application.

### 4.3 Summary

As stated at the beginning of the chapter, the presented architectures are divided into two areas.

The pertinent architectures considered in the area of data analytics are data warehouse, Lambda, and (to some extent) Kappa. These present elements and concepts – such as the ETL process and the asynchronous precomputing of data in batches – that are of relevance for the development of the solution. Their main focus is, however, the provision of preprocessed data to support the execution of data analytics.

The reviewed different software architectures – which can be used to implement data analytics solutions – show a development path. Although there is a clear evolution from monoliths to microservices (over modularization and SOA), all the covered architectural styles present advantages and disadvantages that cause for the former still to be in use.

When comparing software architectures, the criteria utilized are usually considered as context-dependent and related to the background of the stakeholders involved (Svahnberg et al. 2002, pp. 436-439). The criteria used for the comparison in this section reflect the lifecycle of solutions using the architectural approaches covered. They summarize the relevant aspects utilized when describing the characteristics of each software architecture (in particular the advantages and disadvantages) in the corresponding sections in this chapter. These are:

- **Design** criteria cover the conception of the solution and its structure. This considers the granularity of the modules – directly dependent on their size and modularity (as defined in section 4.2.2) – as well as the resulting complexity of the solution (given the number of interconnected elements) and its robustness (e.g. against network failures).



- The **implementation** criteria refer, as the name states, to the factors involved in developing and implementing the solution. This includes the variety of technologies usable, how the solution integrates with other ones in the IT landscape, how independent the development teams are (a factor also influenced by the freedom of choice regarding technologies), and how easy it is to debug the solution. Because of the constant changes during the lifetime of the solution, these factors will also be relevant during the utilization phase.
- **Utilization** criteria refer not only to how the solution is utilized but also to how the system is to be maintained and extended. This includes the scope (typical scale of solutions using the architecture), the maintainability and extensibility (which determine the flexibility of the systems, i.e. how easily can functionalities be replaced or added), and the scalability of the solution, as well as the reusability of the composing modules.
- The **cost** criteria can be considered parallel to all lifecycle phases, with implementation costs – which are relevant mostly at beginning and during big changes – and constantly incurring operational costs.

The criteria categories cover the phases of the waterfall model, a commonly used approach in software development, as described by Adenowo et al. (2020, p. 429) – based on Pfleeger et al. (2010, p. 52).

The resulting comparison of the software architectures covered in this chapter is performed in Table 4-1.

While services have been conceived to be reusable, microservices focus on functionality, limiting their reusability. The combination of SOA and microservices allows for the utilization of agnostic services, improving the reusability in the resulting architecture.

Service-oriented solutions offer a good integration with existing systems in the IT landscape. This is improved by the small size of microservices, which allow complementing business process with individual functionalities.

Complexity is a significant problem in service-oriented approaches. This can be addressed using technological solutions and techniques, one of which being the addition of monitoring systems (Hayashi et al. 2012, pp. 732-737; Daya 2015, p. 28). The

combination of SOA and microservices also improves this issue through the use of hybrid choreography/orchestration approaches.

**Table 4-1: Comparison of software architectures**

		IT architectures			
		Modular monoliths	SOA	Microservices	SOA + Microservices
Criteria					
Design	<b>Granularity</b>	Low	High	Very high	Very high
	<b>Complexity</b>	Low	High	Very High	High
	<b>Robustness</b>	High	Low	Medium	Medium
Implementation	<b>Technology</b>	Limited	Variated	Variated	Variated
	<b>Integratability</b>	Low	High	Very high	Very high
	<b>Development</b>	Bounded teams	Independent teams	Independent teams	Independent teams
	<b>Debugging</b>	Simple	Difficult	Very Difficult	Very Difficult
Utilization	<b>Scope</b>	Enterprise / System	Enterprise / System	Project / System	Enterprise / System
	<b>Maintainability / Extensibility</b>	Difficult	Good	Very good	Very good
	<b>Scalability</b>	Medium-Low	High	Very high	Very high
	<b>Reusability</b>	Difficult	Very good	Good	Very good
Cost	<b>Investment</b>	High	Medium	Medium	Medium
	<b>Operational cost</b>	High	Medium	Low	Low

As consequence of their complexity, service-oriented solutions are more difficult to debug as monolithic ones. Technological methods for improving debugging are in development – for example, Arora et al. (2018) and Zhou et al. (2019).

---

Furthermore, microservices are able to isolate failures, increasing the robustness of the system. This feature is, however, at the same time negatively affected in service-oriented solutions by the possible network problems characteristic of distributed systems (breakdowns and high latency). The situation is worsened in typical SOA systems, where the orchestrator represents a single point of failure.

Maintainability and extensibility are clearly an advantage in service-oriented solutions. These – together with the reusability of the components – also reduce the related operational costs. This is also positively affected by the vendor-neutrality, allowing to freely choose the technology, enabling not only more control over the costs but also increasing the effectiveness of the solution. A trade-off is, however, that the runtime environment (e.g. the Operational and IT Systems Layer) must be able to support the different languages and libraries used (Fowler 2017, pp. 22-23).

While development teams working on monolithic software solutions have to deal with the issue of being bounded by the internal functionality and technology of each module, service-oriented approaches allow developers to work on each service independently, even regarding the choice of technology.

Although the migration of whole systems to a service-oriented configuration can be costly, the investment can be reduced by migrating only some processes, thus extending the remaining monoliths. Monoliths, on the other hand, present the problem of being unable to evolve, causing considerable investments that are necessary in order to upgrade such systems.

Cloud computing is also an option to reduce investments, replacing them by running costs. Although this technology is in principle available for all approaches, it is more suitable for those based on distributed systems and service orientation (Newman 2015, p. 6).

This cloud-based provision of resources also aids scalability, another great advantage of service-oriented solutions. Although monoliths gain scalability through modularity, this is highly limited in comparison to that of their service-oriented counterparts.

Finally, while monoliths and SOA have in common that they were developed to comprehend whole enterprises – or at least one or several systems – microservices focus on individual projects. They can, however, be extended through the combination of SOA and microservices.

Figure 4-6 provides an overview of the advantages and disadvantages as well as an application example of the reviewed software architectures.

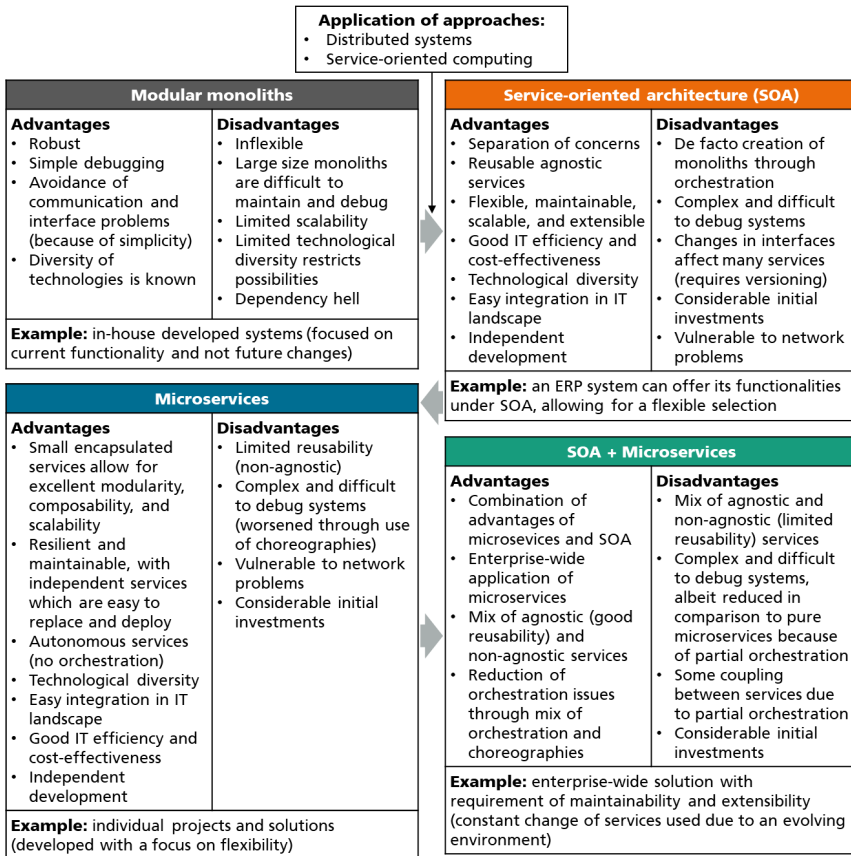


Figure 4-6: Summary of reviewed software architectures

# 5 Proposed Solution: Analytical Microservices

This chapter considers the concepts introduced in chapters 2 and 3 – namely the requirements on a production management software and the features of a possible data analytics-based solution – and proposes a solution combining approaches of SOA and microservices, the so-called “analytical microservices”.

Firstly, the idea of quadruple mirroring is introduced as a way to understand the vision pursued by the proposed solution. Subsequently, the implementation aspects of a data analytics-based solution using SOA and microservices are analyzed and the constituting elements of analytical microservices are introduced.

Finally, a comparison of the proposed solution with existing approaches is performed.

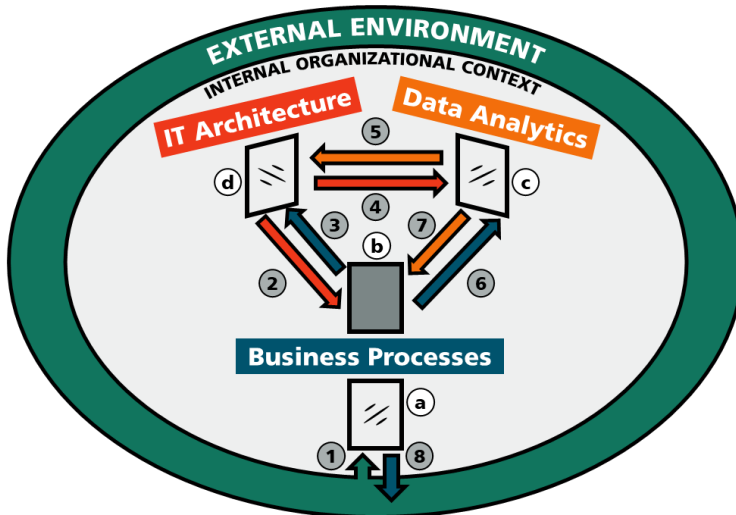
## 5.1 The Quadruple Mirroring Idea

Throughout the previous chapters it was emphasized how vital the capability to adapt is for the concepts introduced:

- An organization must be able to adapt its business processes, matching its internal complexity to the external one in order to manage the latter (law of requisite variety – Ashby’s law).
- Data analytics solutions must adapt to the domain and context where they are applied in order to deliver useful knowledge, as stated by their working principle (especially in the cases of data mining and machine learning).
- The software architecture must be able to evolve with the development of the organization in order to remain effective. Furthermore, the structure of the generated software is not only a mirror image of the organization (Conway’s law), but likewise a

flexible software architecture can influence the structure of the organization, allowing it to be equally flexible (Inverse Conway's Law).

The necessary and continuous adaptation can be looked upon as if the pertained elements were constantly acting as mirrors. Thus, the idea of “quadruple mirroring” – encompassing all three elements simultaneously – came into being. It is depicted in Figure 5-1.



- |  |   |  |
|--|---|--|
| <p>(1) The business processes consider the external complexity.</p> <p>(2) The IT architecture influences the business processes.</p> <p>(3) The IT solutions consider the organizational structure.</p> <p>(4) The IT architecture influences the analytical solutions.</p> | <p>(5) The analytical solutions are implemented according to the IT architecture.</p> <p>(6) The analytical solutions adapt to the business processes.</p> <p>(7) The business processes are influenced by the analytical capabilities.</p> <p>(8) The business processes respond to the external complexity.</p> | <p>a. External mirror of business processes.</p> <p>b. Internal mirror of business processes.</p> <p>c. Mirror of data analytics.</p> <p>d. Mirror of IT architecture.</p> |
|--|---|--|

Figure 5-1: The Quadruple Mirroring Idea

The IT architecture, the employed data analytics, and the business processes each have a mirror. The processes, in their position as the interface to the market, also have an additional mirror pointing to the external environment (market).

Although the “reflections” are numbered, the process should take place continuously and simultaneously. Each reflection has the following meaning:

- (1) The external complexity is reflected into the business processes, which then have to react.
- (2) The constitution of the business process is influenced by the IT architecture: the more flexible the IT architecture, the more flexible the business processes.
- (3) The constitution of the IT solutions – based on the employed architecture – is influenced by the structure of the organization.
- (4) Similar as in (2), the structure of the analytical solutions will depend on the IT architecture employed. This involves both, the way these solutions are implemented (as software-based applications) and their integration with the other IT solutions.
- (5) Analytical solutions are implemented in accordance with the existing IT architecture. This concerns the analytical software and the integration of the corresponding solutions in the IT landscape.
- (6) The constitution of the data analytics-based solutions adapts to the domain and the steadily changing context of the business processes.
- (7) The constitution of the business processes is affected by the analytical capabilities available. For example, good predictive capabilities allow for reducing the dimensions of buffers.
- (8) The business processes respond to the requirements of the external environment (market) with products and services containing the characteristics enabled by the constitution of said processes.

A key role is played by the IT architecture. It determines not only the flexibility of the business processes. It also defines how the analytical solutions can be implemented and integrated into the IT landscape, greatly affecting their functionality and effectiveness.

Furthermore, an evolvable IT architecture will allow the applications based on it – in this case, the analytical solutions – to be equally adaptable to changes.

## 5.2 A Service-oriented Solution

This section will describe the implementation aspects of the proposed solution as a microservices/SOA hybrid and ponder upon the IT efficiency of the emerging approach.

### 5.2.1 Implementation Aspects as a Microservices/SOA Hybrid

Based on the advantages presented by the service-oriented design of software solutions, this is used as a basis for the proposed solution. In particular, the approach using a combination of microservices and SOA was chosen. This work will refer from now on to this combination as **microservices/SOA hybrid**.

To address the features of a possible data analytics-based solution described in section 3.3.3 – which, in turn, addressed the requirements on a production management software summarized in section 2.4.4 – a set of implementation aspects is derived. Using the quadruple mirroring idea as a basis, these aspects adapt the characteristics of the microservices architecture (summarized in section 4.3) in a way that takes into account the advantages and structural requirements of data analytics as well as the needs of production management processes under the challenge of Personalized Production.

The resulting implementation aspects of a data analytics-based solution to support production management software under the challenges of Personalized Production by means of a microservices/SOA hybrid are represented in Table 5-1. These are:

- algorithms as agnostic (reusable) and modular microservices
- models as non-agnostic (specific) and modular microservices
- dynamic generation (training) and performance-based regeneration (retraining) of analytical microservices (models)



- analytical processes as composition of microservices
- flexible selection and exchange of algorithms and analytical models as microservices
- combination of orchestration and choreography to enable synchronous and asynchronous processing
- enhanced performance through usage of microservice scalability
- transfer and distribution of domain knowledge through algorithms and pre-trained models
- transfer and distribution of domain knowledge through pre-built processes
- usage of analytical microservices to support the input phase
- integration in IT landscape through methods of service-oriented architecture

The high granularity and focus on functional capabilities of microservices enable the construction of autonomous components representing the parts of analytical processes, allowing to decompose the latter into as many pieces as functionally required. Furthermore, the possibility to use different technologies in each microservice enables the utilization of a wide range of data analytics.

The emerging analytical microservices can be divided into two main categories (following the constitution of data analytics introduced in section 3.1.1):

- the *algorithms* (the base for the analytical models), which are agnostic and as such highly reusable
- the *models*, which are generated based on the algorithms and are non-agnostic, as they are created within a specific context, limiting their reusability

The approach allows exploiting the reusability of analytical microservices by means of agnostic algorithms and some generic models (e.g. visualizations). How agnostic an algorithm is – and how reusable – will depend on how specific its development was in regard to domain and context.

**Table 5-1: Implementation aspects using a microservices/SOA hybrid approach**

Features of a possible data analytics-based solution (see section 3.3.3)	Implementation aspects using a microservices/SOA hybrid approach									
	Algorithms as agnostic (reusable) and modular microservices	Models as non-agnostic (specific) and modular microservices	Dynamic generation (training) and performance-based regeneration (retraining) of analytical microservices (models)	Analytical processes as composition of microservices	Flexible selection and exchange of algorithms and analytical models as microservices	Combination of orchestration and choreography to enable synchronous and asynchronous processing	Enhanced performance through usage of microservice scalability	Transfer and distribution of domain knowledge through algorithms and pre-trained models	Transfer and distribution of domain knowledge through pre-built processes	Usage of analytical microservices to support the input phase
Possibility to utilize different data analytics techniques	X				X					
Dynamic data-driven generation of specific and accurate analytical models		X	X							
Flexible construction of flows of analytical algorithms/models				X						
Transfer of domain knowledge through pre-trained models							X			
Transfer of domain knowledge through pre-built flows								X		
Ability to select the most adequate available solution, considering different levels of detail and response	X	X		X	X	X				
Ability to integrate with existing production management systems to enable data analytics, continuous learning, and create intelligent processes									X	X
Use of data analytics synchronously and asynchronously					X	X				

---

The proposed solution presents a particularity, as it is partly based on the dynamic generation of microservices through the training of models. This allows not only to create models that adapt to specific context but also to regenerate these models through retraining, changing jointly with the evolution of the manufacturing environment. This continuous adaptation is also supported by the high deployment speed of microservices.

The flows of analytical algorithms/models are built by combining analytical microservices into analytical processes. These are flexible, as they allow adding and replacing analytical microservices, if needed. They enable building complex analytical solutions through the combination of different analytical microservices. Especially relevant is the possibility to build iterative processes, aiding the execution of KDD steps and the realization of special data analytics – for example, reinforcement learning.

Many instances of a process utilizing different forms of the same base model can be created depending on the characteristics of the data – e.g. different product families, production lines, etc. This is possible through the utilization of a service (coordinator) that orchestrates the process and is also reusable.

All of these measures enable the constitution of analytical processes with high accuracy, as they are greatly adaptable to the particular context.

The dynamic generation of analytics, the possibility to freely chose and utilize different technologies, and the flexible modification of analytical processes allow for the selection of the most adequate analytical solution.

The combination of orchestration and choreography enables the execution of synchronous – such as aiding a business process – and asynchronous tasks. The latter can be due to the performance of training processes or simply due to the execution of costly tasks – for example, for lazy learning. The flexibility loss because of orchestration is outweighed by the advantages of explicitly defined processes, such as clarity and reusability. Furthermore, it is always possible to use the analytical microservices reactively (in choreographies), if necessary.

Also reducing the elevated computational costs of analytical tasks is the fact that microservices are naturally scalable. This allows not only choosing the most appropriate execution environment for each analytical function dynamically. It also enables distributing the load through techniques such as partitioning and replication, matching approaches of distributed parallel mining.

One important advantage is the possibility to transfer and distribute domain knowledge. This is achieved by means of the analytical microservices using pre-trained models and algorithms with different degrees of agnosticism (expressed through their structure and parameters), and orchestrating services, which allow transferring the information of processes.

The service-oriented approach allows for a simple integration into the IT landscape. This is enhanced by the utilization of small autonomous analytical functionalities as microservices, which can be used to perform localized extension of business processes – for example, adding a predictive function adapted to the context in order to replace a generic one in the original software. The broad reach of the SOA design allows for a simple enterprise-wide or system-wide integration into the IT landscape.

The bidirectional communication with the manufacturing environment enables the creation of intelligent processes assisted by data analytics. Furthermore, the possibility to receive feedback allows for the utilization of continuous learning and special techniques – for example, instance-based learning.

The integration with other systems enables the utilization of prescriptive analytics to support operative tasks, either by providing a direct and synchronous connection to the process where a “decision” is necessary, or by connecting to other functionalities involved in the decision-making process – e.g. an optimizer.

It is also possible to employ the analytical microservices to support the input phase of the KDD process by assisting in the selection, preprocessing, and transformation of data – tasks also found in the ETL process.

A broad spectrum of analytics is then covered, including all four types of data analytics – descriptive, diagnostic, predictive, and prescriptive. This allows aiding the production management systems in the reviewed aspects (forecasting and decision-making) and supporting analysis.

As explained in chapter 4, the utilization of service-oriented solutions is not without difficulties. These are, however, outweighed by the functional benefits. Furthermore, technical and methodical approaches exist and are being developed to address the network problems, the complexity, and the difficulty to debug present in distributed systems.

Also, building a complex system is not necessarily a problem. Following Ashby's law, a respective internal complexity is required to address the external one.

#### *Completeness of the Implementation Aspects*

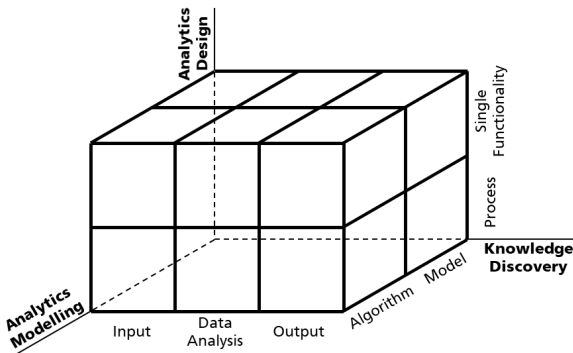
It is necessary to confirm that the derived implementation aspects consider all relevant areas to create a working data analytics solution before continuing with the design.

The first completeness criterion is fulfilled by addressing the derived features of a possible data analytic solution which, in turn, correspond to the requirements on a production management software in order to deal with Personalized Production (as shown in Table 5-1). A further analysis can be made by utilizing sets of criteria to check that the main concepts reviewed in chapters 3 and 4 are being taken in account.

The **analytics design** criteria fulfill a triple purpose. They first address the structural elements of a data analytics solution, either as a single functionality (with all its requirements) or as a process of connected individual functionalities. Secondly, they represent the design pursued in modular software architectures (e.g. microservices), with interconnected functionalities. Lastly, they address the flexibility required to adapt to the changes in business processes (through the use of flexible processes of modular functionalities).

The **analytics modelling** criteria represent the algorithm/model dynamic present in analytical solutions. This is also essential for the creation of analytical functionalities that are able to adapt to the requirements and changes of the underlying business processes.

The **knowledge discovery** criteria address, as the name states, the phases of the knowledge discovery process (as simplified in Figure 3-5). This is not only relevant from the analytical point of view but also for the modular architecture using services and microservices, which are to use the same structure in their own design (with an encapsulated logic doing the analysis and an interface to receive inputs and send outputs to other functionalities).



**Figure 5-2: Completeness criteria for implementation aspects**

As shown in Figure 5-2, the implementation aspects may fulfill several criteria at the same time. The correspondence between the aspects and the criteria are shown in Table 5-2, Table 5-3, and Table 5-4.

Chapter 6 will describe how the components derived from these implementation aspects address the also relevant areas of the SOA Reference Architecture and the data mining phases.

**Table 5-2: Aspects for analytical design and knowledge discovery**

		Analytics Design	
		Single Functionality	Process
<b>Knowledge Discovery</b>	<b>Input</b>	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Usage of analytical microservices to support the input phase</li> </ul> <p><i>Other aspects are the same as by "Data Analysis" in "Single Functionality" (because of possible utilization of data analytics in the input phase)</i></p>	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Usage of analytical microservices to support the input phase</li> </ul> <p><i>Other aspects are the same as by "Data Analysis" in "Process" (because of possible utilization of data analytics in the input phase)</i></p>
	<b>Data Analysis</b>	<ul style="list-style-type: none"> <li>• Algorithms as agnostic (reusable) and modular microservices</li> <li>• Models as non-agnostic (specific) and modular microservices</li> <li>• Dynamic generation (training) and performance-based regeneration (retraining) of analytical microservices (models)</li> <li>• Flexible selection and exchange of algorithms and analytical models as microservices</li> <li>• Enhanced performance through usage of microservice scalability</li> <li>• Transfer and distribution of domain knowledge through algorithms and pre-trained models</li> </ul>	<ul style="list-style-type: none"> <li>• Analytical processes as composition of microservices</li> <li>• Combination of orchestration and choreography to enable synchronous and asynchronous processing</li> <li>• Flexible selection and exchange of algorithms and analytical models as microservices</li> <li>• Enhanced performance through usage of microservice scalability</li> <li>• Transfer and distribution of domain knowledge through pre-built processes</li> </ul>
	<b>Output</b>	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> </ul>	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Analytical processes as composition of microservices</li> </ul>

**Table 5-3: Aspects for analytical modelling and knowledge discovery**

		Analytics Modelling	
		Algorithm	Model
Knowledge Discovery	Input	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Usage of analytical microservices to support the input phase</li> </ul> <p><i>Other aspects are the same as by "Data Analysis" in "Algorithm" (because of possible utilization of data analytics in the input phase)</i></p>	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Usage of analytical microservices to support the input phase</li> </ul> <p><i>Other aspects are the same as by "Data Analysis" in "Model" (because of possible utilization of data analytics in the input phase)</i></p>
	Data Analysis	<ul style="list-style-type: none"> <li>• Algorithms as agnostic (reusable) and modular microservices</li> <li>• Dynamic generation (training) and performance-based regeneration (retraining) of analytical microservices (models)</li> <li>• Flexible selection and exchange of algorithms and analytical models as microservices</li> <li>• Enhanced performance through usage of microservice scalability</li> <li>• Transfer and distribution of domain knowledge through algorithms and pre-trained models</li> <li>• Transfer and distribution of domain knowledge through pre-built processes</li> <li>• Analytical processes as composition of microservices</li> <li>• Combination of orchestration and choreography to enable synchronous and asynchronous processing</li> </ul>	<ul style="list-style-type: none"> <li>• Models as non-agnostic (specific) and modular microservices</li> <li>• Flexible selection and exchange of algorithms and analytical models as microservices</li> <li>• Enhanced performance through usage of microservice scalability</li> <li>• Transfer and distribution of domain knowledge through algorithms and pre-trained models</li> <li>• Transfer and distribution of domain knowledge through pre-built processes</li> <li>• Analytical processes as composition of microservices</li> <li>• Combination of orchestration and choreography to enable synchronous and asynchronous processing</li> </ul>
	Output	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Analytical processes as composition of microservices</li> </ul>	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Analytical processes as composition of microservices</li> </ul>



**Table 5-4: Aspects for analytical modelling and analytics design**

		Analytics Design	
		Single Functionality	Process
Analytics Modelling	Model	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Usage of analytical microservices to support the input phase</li> <li>• Models as non-agnostic (specific) and modular microservices</li> <li>• Flexible selection and exchange of algorithms and analytical models as microservices</li> <li>• Enhanced performance through usage of microservice scalability</li> <li>• Transfer and distribution of domain knowledge through algorithms and pre-trained models</li> </ul>	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Usage of analytical microservices to support the input phase</li> <li>• Analytical processes as composition of microservices</li> <li>• Combination of orchestration and choreography to enable synchronous and asynchronous processing</li> <li>• Flexible selection and exchange of algorithms and analytical models as microservices</li> <li>• Enhanced performance through usage of microservice scalability</li> <li>• Transfer and distribution of domain knowledge through pre-built processes</li> </ul>
	Algorithm	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Usage of analytical microservices to support the input phase</li> <li>• Algorithms as agnostic (reusable) and modular microservices</li> <li>• Dynamic generation (training) and performance-based regeneration (retraining) of analytical microservices (models)</li> <li>• Flexible selection and exchange of algorithms and analytical models as microservices</li> <li>• Enhanced performance through usage of microservice scalability</li> <li>• Transfer and distribution of domain knowledge through algorithms and pre-trained models</li> </ul>	<ul style="list-style-type: none"> <li>• Integration in IT landscape through methods of service-oriented architecture</li> <li>• Usage of analytical microservices to support the input phase</li> <li>• Analytical processes as composition of microservices</li> <li>• Combination of orchestration and choreography to enable synchronous and asynchronous processing</li> <li>• Flexible selection and exchange of algorithms and analytical models as microservices</li> <li>• Enhanced performance through usage of microservice scalability</li> <li>• Transfer and distribution of domain knowledge through pre-built processes</li> </ul>

### 5.2.2 IT Efficiency of the Approach

It is important to consider how the requirements on a production management software are affected by IT efficiency. This is improved in the solution through

- the flexibility of the analytical processes, reducing maintenance and extension costs, and improving the efficacy through adaptation of the solution;
- the dynamic generation and regeneration of models, reducing the continuous adaptation efforts and increasing the effectiveness of models;
- the reutilization of analytical microservices and processes, increasing their cost-effectiveness;
- the possibility to build complex analytics through the simple composition of microservices (analytical processes), increasing the range of possible solutions (and, as a consequence, their effectiveness), enabling the reutilization of existing analytical microservices;
- the reduction of development efforts and the need for expensive specialized resources through the possibility of applying pre-developed analytical microservices and processes with different degrees of domain knowledge (knowledge transfer);
- the utilization of only the required functionalities as microservices, reducing the implementation costs of the service-oriented system;
- the extension of existing system, increasing their maintainability – reducing the related costs – and increasing their effectiveness; and
- the constitution and scalability of the service-oriented solution, which allows employing cloud approaches to reduce investment costs – using an external platform (PaaS) or simply renting the necessary infrastructure (IaaS).

The acquisition of individual functionalities leads on its own to a price advantage, as it allows purchasing only what is required – installing it in an existing platform – avoiding voluminous solutions. The commercialization of microservices and processes (as services) motivates the creation of innovative business and revenue models (e.g. pay-per-use), which can improve the cost-effectiveness of the acquisition of algorithms and models (Bauernhansl et al. 2014a, pp. 27-30; Schatz et al. 2015, pp. 1-15). Although this work

will not perform a deep analysis of commercialization options, it is assumed that these will positively affect the IT efficiency.

Although the application of analytical microservices and processes during the resource-intensive data input phase – selection, preprocessing, and transformation – could increase its effectiveness, such an assessment would be complex, as it is affected by a large number of factors (types of origin systems, data quality, technologies applied, etc.). As a deep analysis is not within the scope of this work, it will be assumed that the performance of such functionalities at least equals that of available tools.

### 5.3 Elements of the Solution

Based on the architectural characteristics of microservices and SOA, on the constitution of analytical solutions, and on the interaction with existing systems, a structure for the proposed solution is conceived.

The structure illustrating the main elements and relationships is shown in Figure 5-3 – a more detailed explanation will be performed in chapter 6.

The following elements can be identified:

- **External data sources** provide the system with the required raw data to perform the analytical tasks (training, testing, execution).
- **External systems**, extended by the analytical microservices, are able to invoke the required functionality using known protocols. Although individual microservices can be invoked, the request is normally handled by the process representing the required functionality.
- **Internal data** represents the storage of extracted data corresponding to replications of the external data sources (if necessary), of operational data required for the internal working of the components (e.g. repositories of analytical microservices and processes), and of execution results of analytical microservices. Storing intermediate results allows it to function as an integration database for sharing data between the

analytical microservices. The database design should be flexible – for example, allowing the dynamic creation of tables or using non structured approaches – but the data format must be standardized to an extent to allow the interoperability between the analytical microservices (e.g. standardizing the denomination of different types of fields).

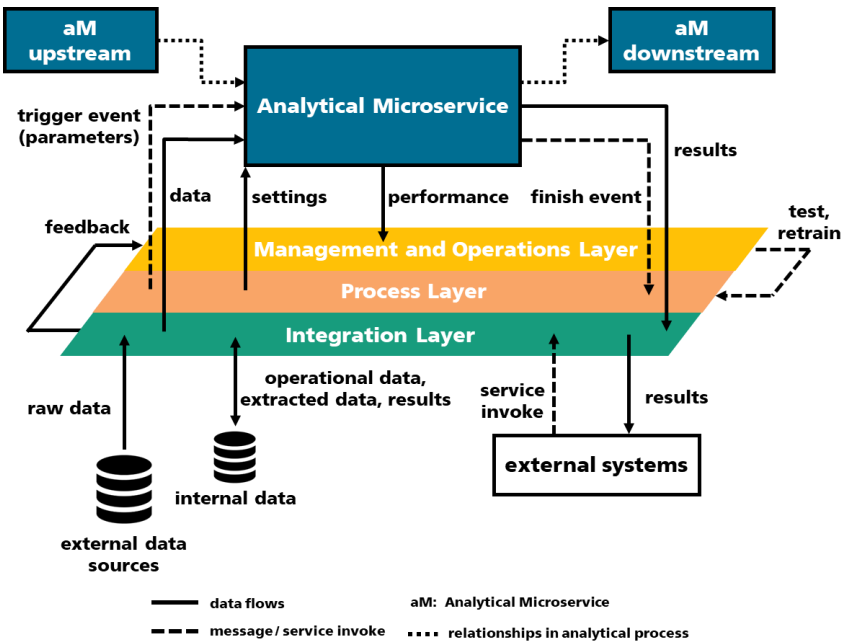


Figure 5-3: Structure of the proposed solution

- The **Integration Layer** handles the interaction with external elements – data sources, systems, user interface – and the internal data flows (including the storage).
- The **Process Layer** contains the services coordinating the analytical processes. The coordination is done via events (messages) used to trigger the execution of analytical microservices – the “trigger event” – and to signalize the finalization of the execution – the “finish event”. This approach – based on the second hybrid solution

in section 4.2.8 – allows extending processes by simply subscribing microservices to the event or adding a new event to the process (in the correct place), indicating in each case the information relevant for the execution (data source, parameters). In this way, a relatively loose coupling with the coordinator is achieved. Furthermore, the coordinating service presents an additional advantage, as it allows choosing the right instance of the process depending on the input data – for example, the same process can use two different models depending on the material type.

- The **Management and Operations Layer** has the functionalities to monitor the technical (e.g. runtime) and functional (e.g. accuracy) performance of the analytical microservices and processes. This allows triggering the retraining (retrain message) of analytical microservices – with the necessary modifications to the corresponding process (e.g. division in new instances) – either manually or automatically. The evaluation can be performed on the results of continuous testing – to verify the validity of the model – or the comparison with feedback from the systems involved.
- The **Analytical Microservice (aM)** represents the autonomous analytical functionality used. Its autonomy allows it to implement different analytical technologies, as long as they are supported by the runtime environment. The standardized interface enables the interoperability with other analytical microservices, forming analytical processes. Input for analytical microservices consists of data from external sources and the results of other analytical microservices (provided through the shared database), parameters (contained in the trigger event), and settings (used during the training phase). The output consists of the results and the performance of the execution.

This structure explains the design of the solution, its components, their functionalities, and the way they are related to each other. It is viewed as an extension of the underlying service-oriented system – for example, extending and complementing the functionalities in a platform. This system will probably take a form based on the SOA Reference Architecture.

The proposed structure presents only relevant and simplified elements, requiring the functionalities of the underlying service-oriented system to function. For example, the information of the Management and Operations Layer can be used to aid in the management of computing resources performed by the Management and Security Aspect of the SOA Reference Architecture.

The relation between the proposed structure and the SOA Reference Architecture will be explained in detail in chapter 6.

## 5.4 Comparison with Existing Approaches

There are a number of analytical software solutions available on the market that can be compared to the proposed solution. The criteria used for the comparison need to be able to provide an overview relevant to the requirements reviewed and derived in the previous chapters. As a result, three sets of criteria must be taken into account.

The **analytical solution** criteria consider the analytical functionalities and capabilities of the solutions to be compared. This pertains the types of analytics that can be used (including the possibility to create complex analytical solutions and utilize processes of analytical functions). Also the adaptability, a main characteristic of solutions based on data mining and machine learning, is considered.

The **IT implementation** criteria, on the other hand, review the aspects regarding the implementation of the analytical approaches as IT solutions. They consider the point of view of resources utilization (including execution and infrastructure requirements) and the integration in the IT landscape.

This **costs** criteria consider the one-time and the continuously accruing costs resulting from implementing and using the solutions compared. It should be taken into account that the costs evaluation can be directly related to criteria in other sets (e.g. maintenance costs can be reduced thanks to good maintainability and extensibility of the solution). Furthermore, costs related to development are organized under one-time accruing costs

---

although they can appear several times during the life of the solution if new developments are necessary (their effect is, however, not continuous).

The point of view of the PPC software, which is also of relevance for the comparison, can be considered as directly dependent on the above presented three sets of criteria. The types of analytics and their adaptability will determine how flexible the related PPC software is and its ability to address specific situations. The IT implementation is critical for executing the necessary analytical functions with an appropriate performance (influencing the response time) and for the integration of the analytical solution with the IT landscape, including the PPC software (taking into account, in this way, a requirement of smart data). The costs criteria are related to the IT efficiency of the solution (alongside the factors described in section 5.2.2).

In this way, the effectiveness and efficiency criteria mentioned in the research question in section 1.3 are addressed.

The comparison criteria derived from the main relevant aspects reviewed in the previous chapters is shown in Table 5-5 (with the corresponding section in parenthesis). Although some could be considered under different categories, the current representation aids the simplicity of the comparison with the avoidance of duplicated criteria. It is, for example, the case of “possible reduction of infrastructure costs through cloud computing”, which could also be considered under the category “resources”.

The comparison is performed in Table 5-6, based on scientific literature, documentation from the providers, and own experience of the author.

The myriad of solutions are organized into categories in order to be manageable. Based on the work of Mikut et al. (2011, pp. 436-441), three categories can be derived: business intelligence solutions, data mining suites, and tailored solutions. A fourth category emerged in the last years to make use of cloud computing and serviced-oriented design: Analytics as a Service (AaaS).

**Table 5-5: Comparison criteria for approaches**

<b>Analytical solution</b>	<b>Types of analytics</b>	<b>Adaptability</b>
	<ul style="list-style-type: none"> <li>• Support of four types of analytics (3.1.1)</li> <li>• Support of complex analytics (3.1.1)</li> <li>• Selection of different approaches and technologies (3.1.1, 3.2.1)</li> <li>• Construction of analytical solutions through composition of analytical functions (3.1.1, 3.1.3)</li> <li>• Equal coverage of entire analytical process (3.1.3)</li> </ul>	<ul style="list-style-type: none"> <li>• Maintainability and extensibility (3.2.3, 4.3)</li> <li>• Continuous adaptation to changes and to the resulting requirements of business processes (3.2.3)</li> <li>• Modularity of analytical functions (3.1.1, 3.1.3, 4.2.2)</li> <li>• Specificity of analytical solution (3.2.1)</li> </ul>
<b>IT implementation</b>	<b>Resources</b>	<b>Integration</b>
	<ul style="list-style-type: none"> <li>• Synchronous and asynchronous execution (3.2.1)</li> <li>• Scalability (4.1.2, 4.3)</li> </ul>	<ul style="list-style-type: none"> <li>• Extension of existing systems in IT landscape (3.2.3)</li> </ul>
<b>Costs</b>	<b>One-time accruing costs</b>	<b>Continuously accruing costs</b>
	<ul style="list-style-type: none"> <li>• Knowledge transfer (3.2.2)</li> <li>• Reusability of analytical functions (3.2.2)</li> <li>• Possible reduction of infrastructure costs through cloud computing (4.2.3)</li> <li>• Implementation costs (3.2.1)</li> </ul>	<ul style="list-style-type: none"> <li>• Maintenance costs (3.2.1)</li> </ul>



**Table 5-6: Comparison with existing approaches**

Comparison criteria	Approaches				
	Business Intelligence Solutions	Data Mining Suites	Tailored Solutions	Analytics as a Service	Analytical Microservices
Support of four types of analytics					
Support of complex analytics					
Selection of different approaches and technologies					
Construction of analytical solutions through composition of analytical functions					
Equal coverage of entire analytical process					
Maintainability and extensibility					
Continuous adaptation to changes and to the resulting requirements of business processes					
Modularity of analytical functions					
Specificity of analytical solution					
Synchronous and asynchronous execution					
Scalability					
Extension of existing systems in IT landscape					
Knowledge transfer					
Reusability of analytical functions					
Possible reduction of infrastructure costs through cloud computing					
Implementation costs (relative amount)					
Maintenance costs (relative amount)					

The trade-off of **BI solutions** – such as *Power BI* (from *Microsoft*) or *Qlik Sense* (from *QlikTech*) – is that the analytical capabilities offered are limited. The focus of such systems is generally on descriptive and diagnostic analytics – heavily depending on visualizations – and simple predictions, usually excluding the utilization of more complex analytics (Kehal 2020, p. 34). The simplicity of the analytics offered allow them to be reusable through easy configuration. However, it highly restricts their capabilities and adaptation possibilities (Bartschat et al. 2019, p. 10).

Though many BI providers offer to extend their software solutions through modularly addable functionalities, such extensions are usually limited to new forms of data presentation. Deeper changes are possible through coding, at the cost of considerable efforts and the need of expert knowledge (van der Lans 2012, p. 55). Even so, this is limited, since the software tool is normally not naturally prepared for such applications.

**Data mining suites** offer the user a wide range of analytics. Common commercial examples include *RapidMiner* (from the homonymous company), *SAS Enterprise Miner* (from *SAS Institute*), and *SPSS Modeler* (from *IBM*). They allow using and reusing modules which are combined in order to build analytical solutions (Chahal et al. 2016, pp. 16-17; Chertchom 2018, p. 49). It also possible, as with the BI solutions, to utilize client/server architectures to manage the processing of analytics, and to use systems that support the data provision, such as data warehouse or implementations of the Lambda architecture (Bartschat et al. 2019, p. 10; Borse et al. 2019, p. 197).

Despite their many capabilities, this type of solutions also presents difficulties. Although many analytical solutions can be built by means of combining functions, this is limited to relatively simple applications. More complex analytical solutions require dealing with intricate configuration and even coding (Bartschat et al. 2019, p. 8). Furthermore, the functions are not independent, as the combination can only be deployed as a whole. As a consequence, the modularity of the functions is impaired. The “flow” of functions can be then thought of as a means to assist in programming the analytical solutions, enabling an easier construction of the basis and a better understanding of the resulting structure.

---

Depending on the complexity of the analytical solutions, maintaining and extending them can become costly (Malkawi et al. 2020, p. 13883).

Similar to BI solutions, many providers offer the possibility to download new functions to extend the capabilities of the suites. These are, however, usually generic operators, limiting as a consequence the possible transfer of knowledge.

Furthermore, although data mining suites allow combining different technologies – for example, R or Python-based applications – this also presents limitations (Walia et al. 2020, pp. 90-92). The suite must provide the required compatibility to use and adapt the components involved.

Both, the business intelligence solutions and the data mining suites, usually offer the possibility to facilitate the access to and the use of their functionalities – for example, by means of interactive and web-based interfaces. Moreover, both of them can be considered as means to create and use monolithic and (to some extent) modular analytical solutions.

The development of **tailored solutions** is considered as a means to create analytical solutions specially adapted to the requirements of the users. They allow applying the technologies and approaches required – for example, R, Python, Java, C++; using the corresponding libraries – creating an analytical solution that can be as complex as needed and well integrated with the existing systems (Bartschat et al. 2019, pp. 8-10). Given the high complexity of knowledge transfer techniques (Mishra et al. 2021, p. 1), tailored solution are very suitable for utilizing them.

Such analytical solutions are normally created as monoliths serving a specific purpose, disregarding aspects of modularity. The trade-off of such solutions is then clear, as their development and maintenance is expensive (Malkawi et al. 2020, p. 13883).

Business intelligence solutions, data mining suites, and tailored solutions can increase their scalability through the use of a solution for distributed processing – for example, *MapReduce* and *Hadoop*, among many others (Stahl et al. 2013, pp. 243-257; Fernández et al. 2014, pp. 380-404). This, however, involves costs associated with the corresponding

effort, which will depend on how well the tool used is prepared to implement such approaches.

**Analytics as a Service** (AaaS) comprise most analytical software solutions making use of service-oriented and cloud computing approaches. They focus on improving the scalability of analytical solutions as well as the way they are provided to the user (Guedes et al. 2006, pp. 36-43; Talia 2013, pp. 98-101). As such, they can actually be considered as a provision of an appropriate execution environment for data analytics. For this reason, Janeczko et al. (2013, p. 9) refers to them as a form of specialized PaaS. Examples of such platforms are *Azure* (from *Microsoft*), *Einstein Analytics* (from *Salesforce*), and the internal data analytics platform developed and used by *Bosch* (Gröger 2018, pp. 5-14). Woo et al. (2018, pp. 2193-2217) proposes using a services-based platform for data analytics, even recognizing the importance of automatically adapting models. However, the analytical models are utilized in a traditional manner (not as services), with the services in the platform performing mainly management tasks (e.g. data collection and performance control), and the approach does not consider the utilization of complex analytical processes. These software solutions can then be regarded as cloud-based forms of data mining suites, sharing many of their characteristics, such as the provision of a wide range of analytics.

AaaS approaches present several limitations. As they prioritize reusability, they are usually forced to standardize the analytical services used (Gröger 2018, pp. 9-10). Furthermore, although the modularity is improved – compared to the data mining suites – this is usually achieved through the utilization of relatively big analytical modules, with the corresponding functional and technical limitations (e.g. scalability). The composition of analytics is frequently limited to the combination of data preprocessing services (ETL) with the analytical services. The usage of these standardized and coarse-grained analytical solutions reduces the specificity and adaptability of the solutions employed.

The capability to extend existing systems in the IT landscape is a problem shared, to different degrees, by all solutions. Business intelligences solutions, data mining suites, and even many Analytics as a Services software solutions focus on providing unidirectional

---

analysis, prioritizing the connection to the data sources, but not the integration of the results in other systems – usually not considering the bidirectional communication in the selection criteria, e.g. Bartschat et al. (2019, pp. 1-14), Walia et al. (2020, pp. 89-94), and Malkawi et al. (2020, pp. 13867-13890). Tailored solutions – which are habitually conceived to extend existing applications – usually require effort to either implement the specific API or to embed the solution in the current system. Gröger (2018, pp. 8-9) proposes using the AaaS platform to aid in the automatization of decision processes, although only expressing it as a necessity, not explicitly explaining how to realize it, nor recommending the integration through service messaging. Extending existing systems with AaaS presents further challenges, as the coarse-grained characteristic and the still limited specificity of the services hampers assisting the underlying system to deal with specific issues and building analytical processes specifically adapted to the problem at hand (e.g. depending on the entry data).

The range of analytics offered is also limited, as the lack of integration with decision processes in the underlying landscape greatly hinders the utilization of prescriptive analytics.

Another issue is the lack of the possibility to continuously adapt to changes and their corresponding effects on the business processes. Additionally to their limited specificity, and although they are able to regenerate the analytical solutions used, data mining suites and AaaS require additional effort to perform deeper structural changes in the analytical solutions. As expressed before, tailored solutions, in spite of their excellent adaptation to specific requirements, are especially costly to adapt or modify. The adaptation capability of BI solutions is lowly graded in comparison to other approaches due to the limitations in the offer of analytical solutions and their generalization – despite being, in many aspects, as configurable as required. This aspect is also negatively affected by the lack of functional integration with the underlying systems, as it hinders comparing the results of the analysis performed to those of the business process assisted (van der Lans 2012, pp. 54-56). Available solutions will then have to rely mainly on manually (re)testing and regenerating models.

Although the utilization of cloud computing offers in all alternatives the possibility to reduce the infrastructure costs, the granularity of service-oriented solutions allows them to make a better use of it.

There are several examples of the utilization of **microservices** in analytical software – for instance, the ones proposed by Stein et al. (2019, pp. 1-3), Jarwar et al. (2018, pp. 112-117), Mouy et al. (2015, pp. 773-779), and Lu et al. (2020, pp. 1-22). These are, however, simply ways to build tailored solutions, which make use of microservices in order to increase the scalability of the solution or facilitate the development of functionalities in independent teams. The microservices provide a certain degree of modularity and improve the maintainability and integration capabilities of the solutions. Nevertheless, this is limited, as they are still dedicated solutions, constructed as an application to address a specific problem, being constituted by non-agnostic microservices thus hindering aspects such as flexibility and reusability. No system for managing the microservices, the processes, and their enterprise-wide (or at least system-wide) integration is created, thereby differing from to the proposed analytical microservices approach.

Furthermore, some providers of production management software offer data analytics solutions to complement their products (Hamedtvasoli et al. 2021, p. 33). These are usually realized as business intelligence solutions – for example, *SAP Business Warehouse* or *abas BI*. In recent time, they have also started initiatives to incorporate more complex analytics, such as data mining and machine learning. An example is the *SAP Leonardo Machine Learning Foundation*. Currently, these approaches take the form of data mining suites or AaaS platforms with limited functionalities. The advantage lies in the good integration with the pairing product, but the offer of data analytics functionalities is restricted (and these often only provide a unidirectional integration with the underlying system).

## 5.5 Summary

A key aspect for the solution described is the proposition of the quadruple mirror idea it intends to implement. The novelty of this approach relies in that it addresses the concepts of adaptability of the business process in an organization, the flexibility provided by the utilized IT architecture, and the adaptability of data analytics-based solutions in a joint manner. This brings the elements reviewed in the previous chapters together instead of considering each one separately, allowing them to profit from each other. The synergy between these concepts provides the organization with the capability to match its internal complexity to the external one, a requirement to deal with the challenges of Personalized Production.

The proposed analytical microservices solution uses its structure to maximize the benefits of data analytics by supporting analytical approaches of all four types, enabling high complexity techniques (including the creation of analytical processes) and promoting their learning capabilities. Knowledge transfer is supported by the possibility to utilize pre-trained models and pre-built analytical processes.

At the same time, several challenges of data analytics are addressed (enumerated in section 3.2). The microservices architecture allows for an easy replacement of analytical functionalities and thus for a selection of the best solution for each case. Furthermore, the flexibility of the solution is a key factor in reducing the costs associated with utilizing data analytics. Moreover, the integration capabilities of the microservices approach enables an easy and close integration with other systems and the business processes requiring the analytical support. In this way, the basis for smart data is provided.

These characteristics separate the solution from other approaches available in the market. Existing solutions usually implement aspects of data analytics only partially (as detailed in section 5.4), depending on their intended utilization. Furthermore, the effort and costs associated with adapting such solutions (if the required adaption is even possible) are generally high.

From an architectural point of view, the proposed solution makes full use of the capabilities of microservices described in chapter 4. The approach even extends the flexibility provided by the architecture by allowing the dynamic generation of microservices in a simple manner (without having to program them). This is done through the creation of analytical models as the result of the training of algorithms, exploiting in this way the learning capabilities of data analytics. Furthermore, the combination of SOA and microservices allows for the creation of understandable analytical processes, addressing the complexity problem associated with pure microservices. The functionalities of the services acting as coordinators (as described in section 4.2.8) are extended, enabling the asynchronous execution of steps in the analytical process (a requirement of many analytical techniques) and the transfer of pre-built analytical processes.



# 6 Constitution of Analytical Microservices and Their Environment

An introduction to the concepts of the proposed solution was performed in chapter 5. Chapter 6 will focus on extending the description of the solution by explaining the mechanics behind it.

As mentioned in chapter 1, the construction of a reference model was chosen as the IT artefact to serve as the basis for the explanation of the proposed approach. Such models can also be considered as the first step in order to develop future IT implementations of the solution (Becker et al. 2009, pp. 181-183), allowing to gather and address the requirements (Becker 2004, p. 74) and thus reduce the associated risks (Becker 2004, p. 83).

## 6.1 The Modelling Language

The modelling language chosen for illustrating the key elements of the reference model is the Unified Modelling Language (UML). This not only provides the possibility to model the required elements – through the use of components and sequence diagrams (Miles et al. 2006, pp. 11-12) – it is also the de facto standard in the software industry (Nugroho et al. 2008, p. 90). This status ensures the interpretability and usability of the reference model. Furthermore, other languages seen as alternatives to UML (e.g. SysML) are often actually extensions of it (García-Borgoñón et al. 2014, pp. 110-111).

It was decided to use the current version of UML: UML 2.0. Among the advantages of extending and improving the base language, it is worth mentioning that version 2.0 was created with the design of Platform Independent Models (PIMs) in mind, which allows for “models that capture the system in a generic manner that is divorced from concerns such as implementation language and platform” (Miles et al. 2006, p. 10).

## 6.2 Structure of the Proposed Solution

This section will describe the structure of the proposed solution. The description will be divided in two: the analytical microservice and the analytical environment in which the former is to be used.

### 6.2.1 Analytical Microservices

The constitutive elements of the proposed solution were introduced in section 5.3. The suggested layers serve the purpose of supporting the main elements: analytical microservices (aMs) – including their utilization and lifecycle – and the processes resulting of their composition.

Based on the concepts reviewed in the chapters 3 and 4, and on those described in the ISO/IEC 13249-6:2006 (pp. 11–17), it is possible to derive the required inputs and outputs of the designed analytical microservice. The resulting data flows – depicted in Figure 6-1 – are

- the **settings** (input) used during the training phase of the aM;
- the **parameters** (input) used during the application of the aM;
- the **input data** (input), consisting of external data (from external data sources and systems) and, if necessary, data originated in the previous aM in the analytical process (aM upstream) stored internally (internal data);
- the **results** produced (output), which can either be stored internally to be used by the following aM (aM downstream) or be communicated to the invoking system; and
- the **performance** (output) evaluating the results of the training, test, and application phases.

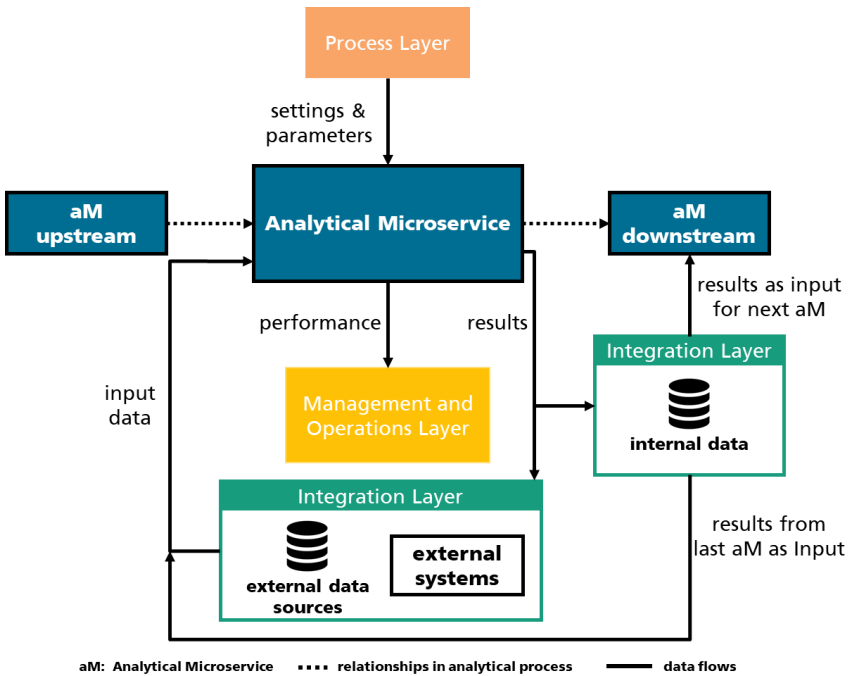


Figure 6-1: Data flows and elements of an analytical microservice

The settings and parameters allow the reutilization of one aM for diverse purposes in different contexts and situations.

It is possible to distinguish between three types of aMs:

- **trainable algorithms**, which provide the basis for the creation of data mining models
- **applicable models**, utilized when running the intended analysis
- **pre-trained models**, which must be adjusted in order to be applicable

As explained in chapter 3, algorithms in data mining approaches are trained in order to generate models. This is, however, only one way to create models. The applicable models in the intended solution cover a wide range of logics: from simple mathematical formulas,

through statistical constructs, to full data mining models; being as specific and complex as necessary and possible. This allows covering an ample number of different use cases.

The pre-trained models follow the idea of using previously created models, which are to be adapted to be utilized in the intended context and purpose. The concept is based on the transfer learning approach, allowing the transfer and reutilization of domain knowledge, with the benefits enumerated in the previous chapters. The adaptation is performed by means of the settings of the corresponding aMs.

Both the proposed interface for the aMs and the flexibility of their internal logic are intended to cover a wide range of applications in a standardized manner. The following characteristics also support this purpose:

- Both algorithms and models can be used to conform analytical processes by composing the respective aMs. This allows the creation of complex structures for advanced analytics.
- In contrast to standard microservices-based solutions, the proposed approach supports the dynamic instantiation of aMs. This enables offering analytical solutions highly adapted to each problem and able to evolve with the changes in the context. Different models can be instantiated, for example, through training with diverse data sets and settings (e.g. to adapt to various products or machines with significantly dissimilar production conditions).
- The flexibility to build analytical processes – comprised of dynamically instantiable aMs – as required by each scenario constitutes the basis for one of the main advantages of the solution: the ability to easily select and utilize the algorithm or model that best suits each situation.

Not only the pre-trained models, but also the distribution of algorithms – and even of pre-built analytical processes – allow for an easy transference of domain knowledge.

### 6.2.2 The Analytical Environment

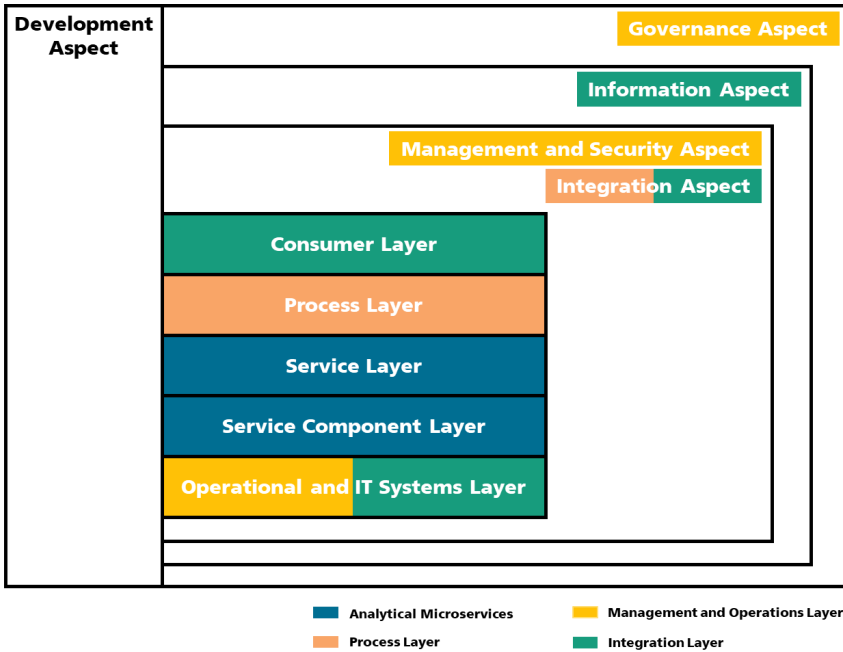
As stated before, analytical microservices and processes exist and run within an environment that allows their correct execution and integration with the user systems. This analytical environment consists of the three layers introduced in section 5.3:

- Integration Layer
- Process Layer
- Management and Operations Layer

These layers, together with the analytical microservices, fulfil the requirements of both aspects of the solution: the utilization for analytical purposes and its microservices-based character. As such, a relation can be found between the proposed elements and the reference architectures that were used as a basis. An example of how the elements can be linked to the SOA Reference Architecture (SOA RA) (described in section 4.2.6) is shown in Figure 6-2.

The **Integration Layer**, as its name indicates, is responsible for managing the integration of the proposed solution, allowing the interaction and data exchange both internally (between the components of the solution) and externally (with external actors). As such, it encompasses the Information Aspect, the Integration Aspect, the Consumer Layer, and parts of the Operational and IT Systems Layer – those referring to the underlying communication infrastructure – of the SOA RA.

The **internal data** can be considered under the influence of the Integration Layer. This allows the intermediate storage of data that, among other functions, enable the communication between aMs in analytical processes (by storing the results to be passed to next step). It also supports the data storage needs of the other layers and can, if necessary, act as part of the ETL process.



**Figure 6-2: Relation with the SOA Reference Architecture  
(based on ISO/IEC 18384-2:2016)**

The external communication functions have two main objectives: the connection with external data sources and the communication with external systems. The difference between them is however minor, as external systems may also be, depending on the circumstances, data sources, and a bidirectional communication with external data sources may be necessary in order to report results.

The act of connecting with the data sources can be limited to simply mapping fields or be more complex, involving ETL functions. The integration with an existing data warehouse should be possible – and even advisable if the data volumes require it.

The term *external systems* not only refers to legacy systems making use of the analytical environment but also to all forms of interaction with users (e.g. the control interface).

The **Management and Operations Layer** embodies the administrator of the analytical environment. With this function it encompasses the Governance Aspect, the Management and Security Aspect, and the remaining parts of the Operational and IT Systems Layer of the SOA RA.

From a technical point of view, it manages the execution and resources utilization. As a result, an important task is governing the scalability of the system, which involves replicating and distributing instances of analytical microservices and handling the asynchronous execution of aMs.

A task specific to the proposed solution is the monitoring of the functional performance of the analytical microservices and processes. The components of the layer can decide on the adequacy of the analytical approach based on the reported performance of the running phases – training, test, and application – and/or via the comparison with the reality provided by the external systems and data sources (e.g. accuracy of predictions). As a result, it can trigger actions such as the retraining of models (which can include a change in the settings), changing the conditions governing the circumstances under which each analytical microservice and process is to be used (e.g. assigning analytical approaches to material types depending on which is a better fit), or swapping and adding new analytical microservices and processes to replace or extend the existing ones. A continuous monitoring is necessary in order to keep up with changes in the manufacturing environment.

A *control interface* must be provided on the external side of the environment in order to allow monitoring by users (the communication is provided through the Integration Layer).

The **Process Layer** is responsible for managing the analytical process created through the composition of the analytical microservices. As such, it encompasses the Process Layer and parts of the Integration Aspect – those responsible for the administration of events – of the SOA RA.

A main component of the layer is the *Process Engine*. On the operative level, it contains and instantiates the coordinating services responsible for controlling the execution of the corresponding analytical process via events. On a management level, it decides, based on

the parameters passed by the invoking system, which analytical processes should be executed.

The instantiation of the coordinating services is made based on the process description. This can be found in the *Process Repository*, which is responsible for storing and managing the analytical processes.

The event-based management is supported by the *Message Broker*, which enables the subscription of analytical microservices and processes to events, being also responsible for communicating the occurrence of such events (thus starting the actions of the subscribed microservices and processes).

**Analytical microservices** encompass the Service Layer and the Service Component Layer of the SOA RA.

Important components of the analytical microservices area are: the *Analytical Microservices Operation*, responsible for the execution of the aMs (acting as a container); the *Algorithms Repository*, used for the storing and management of algorithms; and the *Models Repository*, used for the storing and management of models.

## 6.3 Analytical Processes

This section will explain the main aspects of the analytical processes to be built within the proposed solution. Additionally, usual and particular cases are addressed.

### 6.3.1 Fundamentals of Analytical Processes

Analytical processes are a crucial element of the proposed solution. Analytical microservices must, in order to be executed, be contained within a process. This is due to the Process Layer coordinating the execution, making decisions based on the parameters, and balancing the utilization of resources. Even the utilization of only one analytical microservice requires the creation of a trivial process with only one step.



Such trivial case is depicted in Figure 6-3. The intended standard way of utilization is also shown: supporting a business process running in an external IT system. A step of the process requiring the assistance of a data analytics-based function proceeds to invoke the analytical process by means of the corresponding protocol – e.g. SOAP or REST (Dowall 2018, pp. 109-112) – transmitting the corresponding parameters.

After the end of the analytical process the results are communicated to the invoking step in order to proceed with the business process.

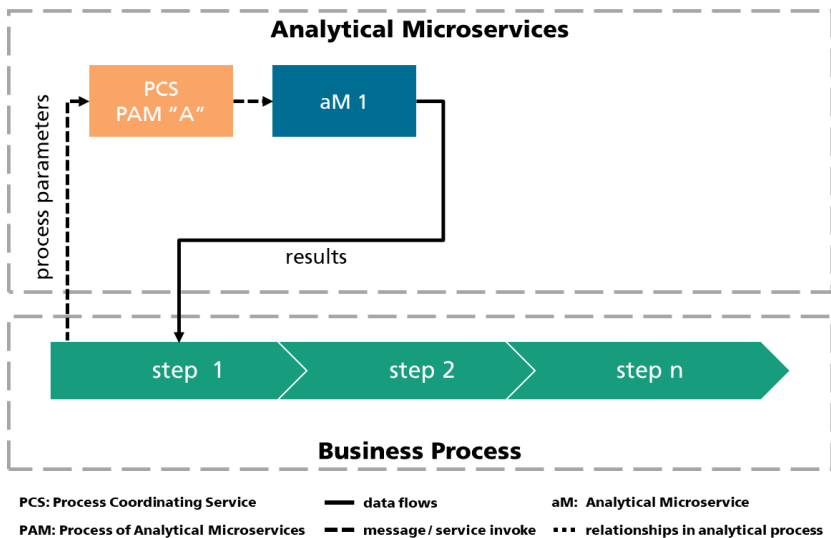


Figure 6-3: Trivial analytical process

It is also possible to invoke an analytical process outside of a business process – the most common example would be the ad hoc utilization by users. However, the procedure described differs very little: mainly the invoking agent must be replaced. As stated before, the external invoking agents will be generally referred to as external systems within this chapter.

A more detailed view of how an analytical process is invoked and executed is provided in Figure 6-4. In order to provide a detailed description, the specific object representing the composition of analytical microservices will be referred to as *Process of Analytical Microservices* (PAM). *Analytical process* is considered a comprehensive term referring to the whole procedure (managing the invocation, selecting a PAM, etc.), of which the executable PAM is a part.

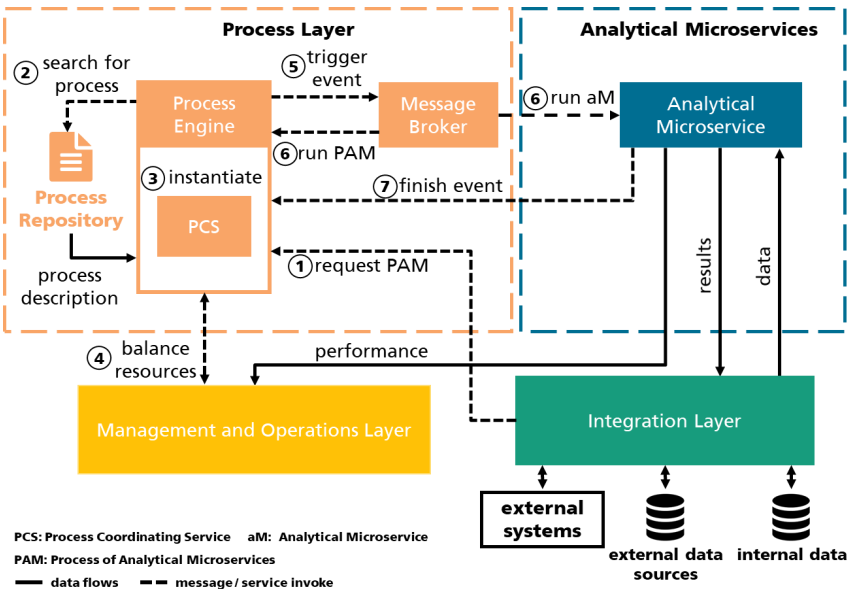


Figure 6-4: Detailed execution of an analytical process

The invocation of the process begins with a request placed through the Integration Layer calling for the execution of a PAM. This request is received and managed by the Process Layer, more specifically by the Process Engine. In order to fulfil the request, a coordinating service responsible for managing the execution is instantiated – this service will be referred to as *Process Coordinating Service* (PCS). The service is based on the process description contained in the Process Repository, which stores the steps of the PAM together with the

corresponding events and parameters. Additional information about the PAMs to be used in the decision-making process of the Process Engine – such as area of validity of each PAM – can also be stored in the Process Repository.

After instantiating the PCS, and prior to starting the execution of each aM, the Process Engine communicates with the Management and Operations Layer in order to coordinate (balance) the utilization of resources – thus managing the scalability of the solution. This balancing can result, for example, in a replication and distribution of the instances of the aM (controlled by the Management and Operations Layer) or in the asynchronous execution of the aM (controlled by the Process Engine).

The PCS then proceeds with the execution of the PAM. The events representing the different steps of the PAM are triggered and communicated to the Message Broker. It is possible to transfer parameters – such as the ID of the calling PAM – within the event. Another possibility is to use the internal data to pass parameters.

Upon receiving the messages of the events, the Message Broker communicates their occurrence to the subscribed aMs, triggering their execution. It is even possible for a PAM to subscribe to an event, thus triggering a sub-PAM within the main PAM.

For its execution, each analytical microservice will make use of the transferred parameters and the required data (from external systems/sources or from the previous aM), which are requested through the integration layer. After the execution, the aM proceeds to communicate

- the finalization to the Process Layer (finish event),
- the performance of its execution to the Management and Operations Layer, and
- the results of its execution to the Integration Layer (in order to be passed to the corresponding consumer).

The utilization of the PCS enables a loose coupling between the analytical microservices while providing a clear overview over the process, allowing its easy management (following the concepts in section 4.2.8). It is only necessary for the aMs – and eventually the PAMs – to subscribe to the corresponding event in the Message Broker. This

component also supports the loose coupling by enabling the asynchronous communication between the PCS and the aMs.

### 6.3.2 Possible Configurations of Analytical Processes

As stated before, the process in Figure 6-3 represents the simplest form of a PAM constituted by a singular analytical microservice. This is, however, usually not the case.

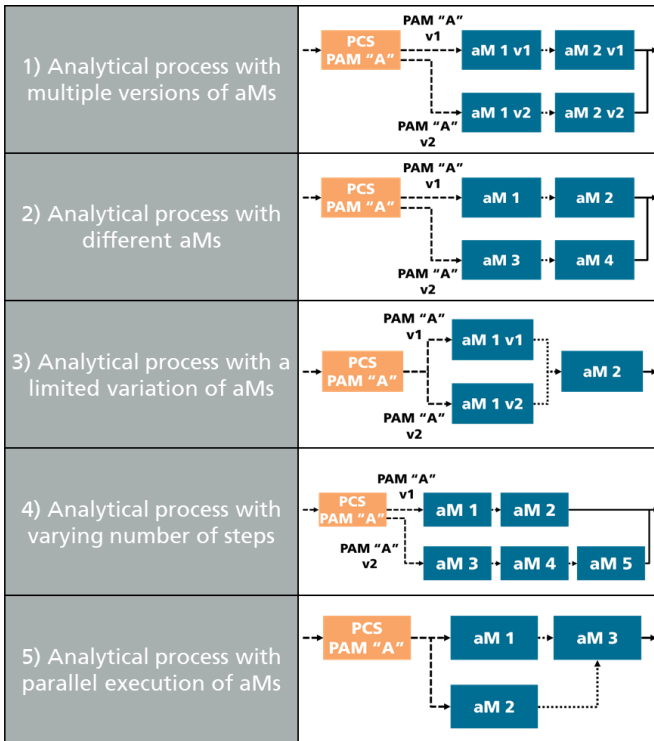
As explained in chapter 3, data analytics techniques are to be viewed as processes composed by several analytical functions which, in the proposed solution, take form through analytical microservices and processes. This form of constitution serves the following purposes:

- the coverage of the different steps of the KDD process, with aMs supporting, for example, the data selection and preprocessing steps
- the creation of complex analytical solutions composed by aMs representing different analytical approaches building upon one another (e.g. elimination of outliers followed by clustering), also fulfilling the requirements of special analytical solutions (e.g. hierarchical learning and reinforcement learning)

Furthermore, an analytical process may require the utilization of diverse aMs depending on the application context in order to increase its accuracy or ensure its usability. Such is the case, for example, when the material disposition presents different behaviors depending on the type of article considered or when machines perform differently based on their make, age, the raw materials employed, etc. The association between the application context and the corresponding analytical approaches can be determined using the parameters provided by the former (usually the IT system supporting the business process). When selecting the appropriate analytical approaches, the Process Engine could work in two ways:

- a PAM exactly corresponding the context parameters is chosen
- a PAM covering the specified context parameters (among others) is chosen

In the second alternative, the ultimate decision regarding which aMs to use will be managed by the PCS. This approach allows ensuring the clarity of the analytical processes, being able to either create PAMs for each individual situation, or to create different PAMs to cover generic situations within an analytical process, being able to extend them to consider particular cases. Based on the second alternative, several configurations of analytical processes can be considered, as depicted in Figure 6-5.



**aM: Analytical Microservice**

**PCS: Process Coordinating Service**

**PAM: Process of Analytical Microservices**

**Figure 6-5: Overview of possible configurations of analytical processes**

An **analytical process with multiple versions of the aMs** can be used when the same fundamental approach is to be used but with small changes based on the particular situation. It is, for example, the case of aMs employed to predict the yield of a machine used in different lines: although the algorithms may be the same, they are trained using distinct data sets for every line – as each can behave differently based on factors such as the other machines used, the operators, the materials, the age of the machines, etc. How such configuration could look like is illustrated in Figure 6-6, with the PCS deciding which branch of the PAM to use depending on the process parameters (e.g. the ID of the work station or the line).

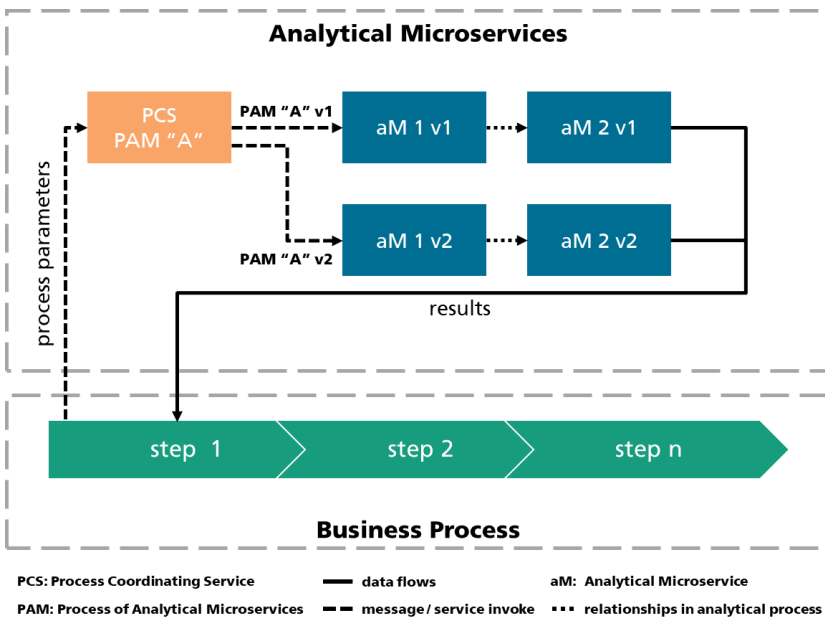


Figure 6-6: Analytical process with multiple versions of aMs

When the discrepancy between the foreseen situations is too great, it is necessary to use an **analytical process with different aMs** according to the application context. It is, for

example, the case of several lines fulfilling the same purpose but using machines from different manufacturers: because of the difference in the internal working of the machines they may require different analytical approaches to predict the behavior of the lines. A model of this configuration is illustrated in Figure 6-7.

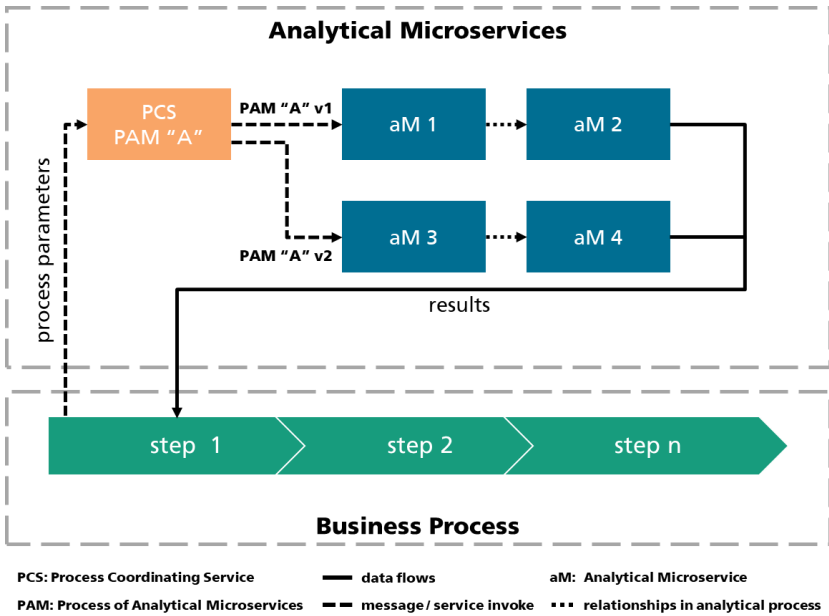


Figure 6-7: Analytical process with different aMs

However, it is not always necessary to create processes with completely different branches for each situation. It is possible to create **analytical processes with a limited variation of the aMs**, subjecting the utilization of only one or several aMs to the parameters considered. It is the case, for example, when materials differ in the way they are procured. For the purpose of calculating the throughput time of a product, each material type may require a different analytical approach in order to calculate the corresponding procurement time. However, as the subsequent manufacturing process may be the same,

all materials may share one analytical approach for calculating the corresponding time. Although in Figure 6-8 a simple example is illustrated in which another version of the aM is utilized, it would also be possible to use completely different aMs. In order to execute the divergent aM, it is only necessary for the PCS to use the corresponding event, which will be different for each aM in the PAM.

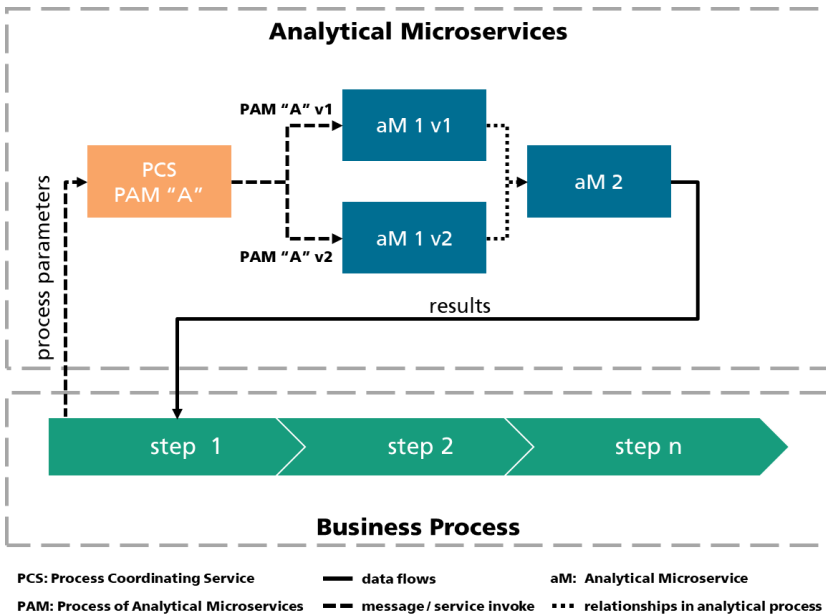


Figure 6-8: Analytical process with a limited variation of aMs

If the analytical approach for each situation needs a different number of analytical functions, the utilization of an **analytical process with a varying number of steps** is required. It is the case, for example, when the data quality in different production lines differ: while the input data may be directly usable in some lines, that from other ones may require the application of data preparation aMs. A model of this configuration is illustrated in Figure 6-9.



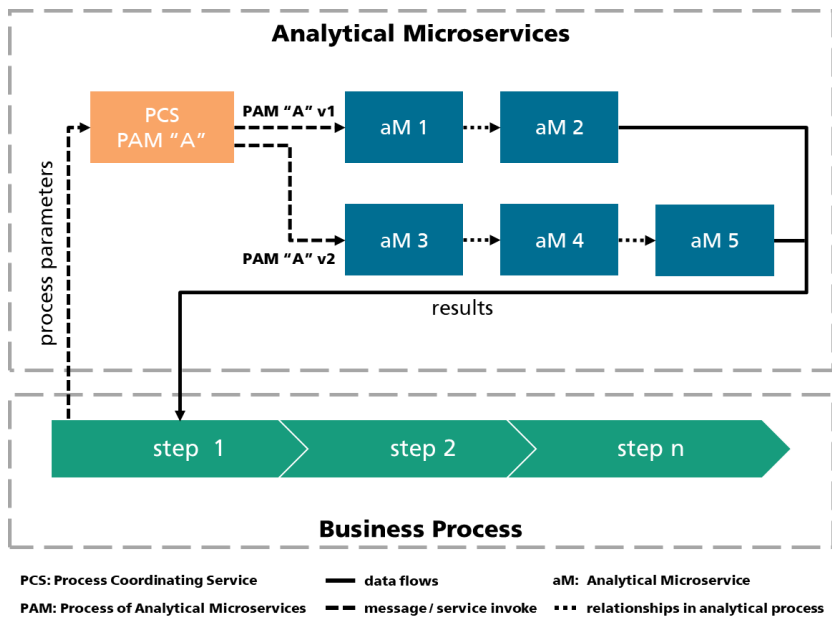


Figure 6-9: Analytical process with varying number of steps

A step of an analytical process is not composed necessarily by only one aM running at a time. It is possible to build **analytical process with parallel execution of aMs**. Such is the case, for example, when the estimation of the throughput time requires the separate calculation of the operation and the inter-operation times, which are then utilized in a subsequent aM. A model of this configuration is illustrated in Figure 6-10. In order to execute the parallel aMs, it is only necessary for the PCS to use the one event to which the aMs are associated.

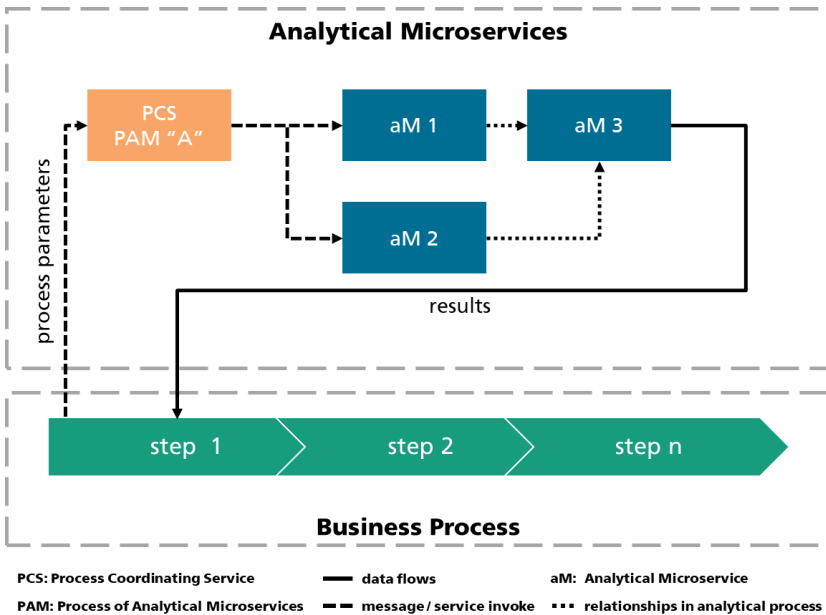


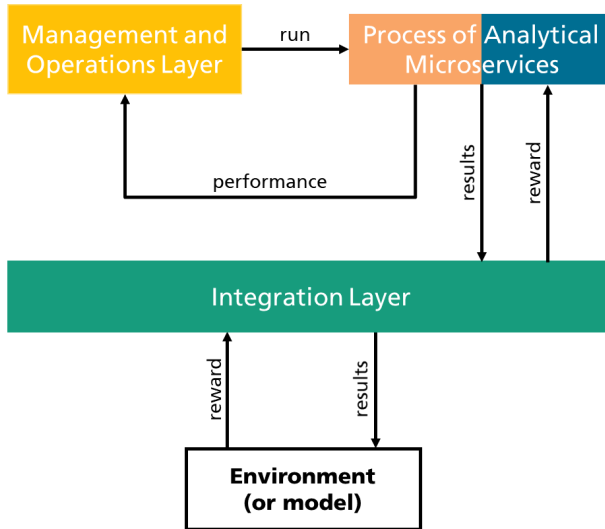
Figure 6-10: Analytical process with parallel execution of aMs

### 6.3.3 Further Considerations and Particular Cases

As explained in chapter 3, one important part of data analytics as processes is the possibility to perform iterations. This is feasible in the solution by means of the Management and Operations Layer, which is able to trigger iterations on its own or on request by the aMs. One important application of this capability is the possibility to retrain models in order to fine-tune them or to adapt to changing conditions.

A particular case of the retraining process is constituted by the iterative procedure required by a reinforcement learning approach. How such a process could be modelled in the proposed solution is illustrated in Figure 6-11. The Management and Operations Layer would have, aside from the initial run, the important function of triggering the execution (retraining) of the PAM based on the *performance* reported by the corresponding

analytical microservice/process, which would contain the result of the evaluation of the reward. This second function, however, is only necessary if the aM/PAM is not capable of triggering its own execution based on the reward assessment itself.



**Figure 6-11: Iterative process for reinforcement learning**

As stated in the previous chapters, the capability to run the analytical microservices synchronously and asynchronously is necessary to enable scalability and to use particular analytical approaches (e.g. instance-based learning). PAMs in the solution are capable to do this by managing the scheduling of the triggering events (in coordination with the Management and Operations Layers).

Also relevant for the scalability is the possibility to replicate and distribute the instances of the aMs. This would mean that the PCS must be aware of the different instances while managing the execution of the PAM. This is possible through the coordination with the Management and Operations Layer.

### 6.4 Components and Interfaces

In this section, a comprehensive description of the main interactions between the components introduced will be given. Within the UML language, these interactions are known as *interfaces*.

A component diagram allows (within the UML language) the representation of the components and their interfaces. A simplified version of these diagram applied to the solution can be seen in Figure 6-12. A detailed diagram illustrating each interface can be found in Appendix A. It is important to take into account that many acts of internal communication normally managed by the Integration Layer are not represented even in the detailed diagram in order to support readability.

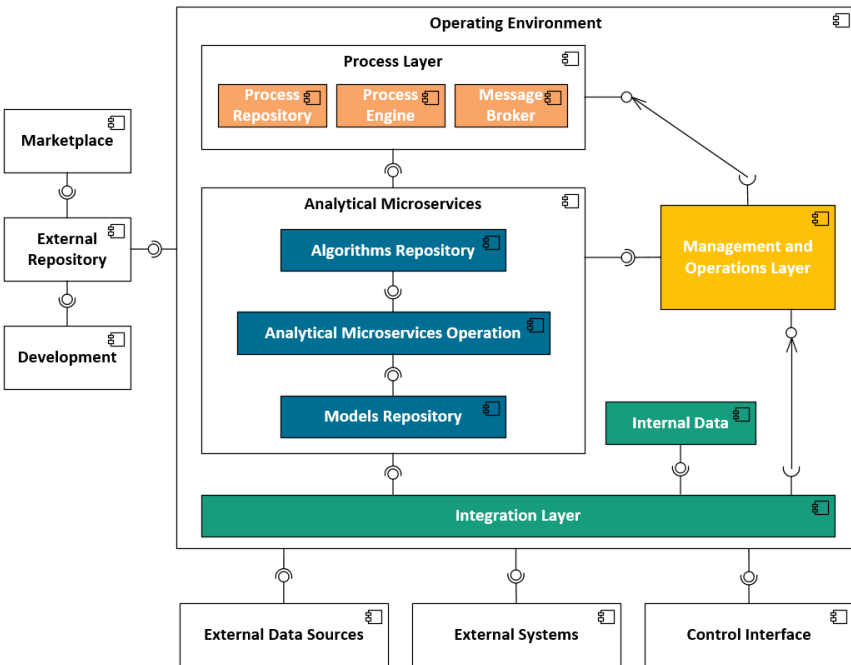


Figure 6-12: Simplified component diagram of the solution

Additionally to the already mentioned components, the external sources where aMs and PAMs can originate are introduced in Figure 6-12: a *marketplace*, from where they can be acquired and downloaded; and the *development*, representing the corresponding programming environment and tools. An *external repository* fulfils the function of intermediate storage before importing the elements.

The interfaces to be considered are (detailed in Appendix A, with their technical names in parenthesis):

- **Request Configuration of PAM** (*RConfigurePAM*) refers to the request from the control interface to the Management and Operations Layer (over the Integration Layer) to configure a PAM (which can refer to the PAM itself or its composition).
- **Configure PAM** (*ConfigurePAM*) continues the above-described request. It enables the Management and Operations Layer to manage the configuration of the PAMs and aMs through the Process Engine. Changes to the composition of the PAM may trigger the necessity to change the association of aMs to events.
- **Publish aM** (*PublishAM*) triggers, on request by the Management and Operations Layer, the publishing of analytical microservices – the association to the corresponding events – in the Message Broker.
- **Publish PAM** (*PublishPAM*) triggers, on request by the Management and Operations Layer, the publishing of a PAM in the Message Broker (allowing a PAM to invoke another).
- **Search PAM** (*SearchPAM*) allows the Process Engine to look up PAMs and their characteristics in the Process Repository.
- **Request Run PAM** (*RRunPAM*) refers to the request by an external system (users or IT systems) or the control interface to the Management and Operations Layers (over the Integration Layer) to run a PAM. To *run* a PAM may refer to the following actions: train, test or apply. It may involve the PAM as a whole or specific aMs contained within it.
- **Run PAM** (*RunPAM*) allows the Management and Operations Layer to request the Process Engine to run (train, test or apply) a PAM. The request may also originate in the Message Broker if a PAM is associated to an event.

- **Balance Resources** (*BalResources*) enables the Process Engine to coordinate the resources utilization with the Management and Operations Layer. This allows managing the scalability of the systems and scheduling the execution events in the most convenient manner.
- **Trigger Event** (*TriggerEvent*) enables the Process Coordinating Services (contained in the Process Engine) to communicate the triggering events of the PAM to the Message Broker.
- **Run aM** (*RunAM*) allows the Message Broker to communicate the Analytical Microservices Operation component the occurrence of a triggering event, prompting it to run the corresponding aM.
- **Search Algorithm** (*SearchAlgorithm*) allows the Analytical Microservices Operation component to look up aMs constituted by trainable algorithms in the Algorithms Repository, in order to run them or access information about them.
- **Search Model** (*SearchModel*) allows the Analytical Microservices Operation component to look up aMs constituted by models in the Models Repository, in order to run them or access information about them.
- **Request Data** (*RData*) allows the Analytical Microservices Operation component to request the Integration Layer for the data (from external or internal sources) required for the execution of the aMs. It can also be used by the Management and Operations Layer in order to assess the accuracy of the current PAMs by comparison with the results contained in the systems involved (feedback).
- **Access External Data** (*AccessDataE*) is utilized by the Integration Layer in order to access and transfer the data required from external data sources.
- **Access Internal Data** (*AccessDataI*) is utilized by the Integration Layer in order to access and transfer the data required from internal data sources.
- **Report Performance** (*RepPerform*) allows the Analytical Microservices Operation component to report the performance measured during the execution of an aM.
- **Report Finalization** (*RepFinish*) allows the Analytical Microservices Operation component to send the finish event to the Process Engine, marking the end of the run of an aM.

- 
- **Report Results** (*RepResults*) allows the Analytical Microservices Operation component to communicate the results of the run of an aM to the Integration Layer which then sends them to the corresponding consumer.
  - **Communicate Results** (*ComResults*) is used by the Integration Layer in order to communicate results to the external systems – by either transferring them to the invoking system or by means of storing them in the corresponding data sources.
  - **Save Results** (*SaveResults*) is used by the Integration Layer in order to store intermediate results in the internal data component.
  - **Save Model** (*SaveModel*) is used by the Management and Operations Layer to trigger the storing of a trained model with acceptable accuracy in the Models Repository (after being temporarily stored in the Analytical Microservices Operation component).
  - **Request Status** (*RStatus*) allows the control interface to request information about the working status of the solution (e.g. performance of the PAMs) from the Management and Operations Layer (over the Integration Layer).
  - **Request Information about aM** (*RInfoAM*) enables the Integration Layer to request information about the interface of an aM in order to manage its integration (e.g. for the connection to the data sources).
  - **Download aM** (*DownloadAM*) represents the process of downloading an aM from the marketplace and adding it to the external repository.
  - **Download PAM** (*DownloadPAM*) represents the process of downloading a PAM from the marketplace and adding it to the external repository.
  - **Save Developed aM** (*SaveDevAM*) represents the process of saving a developed aM into the external repository.
  - **Save Developed PAM** (*SaveDevPAM*) represents the process of saving a developed PAM into the external repository.
  - **Import aM** (*ImportAM*) allows importing an aM from the external repository (using the Integration Layer to manage the request). This interface can be triggered by the configuration of a PAM.

- **Import PAM** (*ImportPAM*) allows importing a PAM from the external repository (using the Integration Layer to manage the request). This interface can be triggered by configuration of a PAM.

## 6.5 Description of the Main Utilization Forms

In this section the main utilization forms of the proposed solution will be described. The representation form chosen is the sequence diagram (provided by the UML language), which shows the participants, the way they interact as messages (with arrows), the main parameters of such messages (within parenthesis in their description), and the processes (with activation bars).

First of all, it is important to consider the **configuration** sequence, depicted in Figure 6-13. This sequence allows tasks such as changing the composition of the PAMs (aMs involved and their order) and the settings and parameters utilized (e.g. for the area of application). These changes may lead to a *publish* message to accommodate the aMs to the corresponding events in order to match the intended execution order in the PAM.

The configuration sequence is intimately related to those for training and testing as, depending on the changes, it may be necessary to trigger a training process and/or test the accuracy of the models.

The configuration will normally be started as a request to the Management and Operations Layer (*M & O Layer* in the diagram), although it is conceivable for this layer to start it itself as part of continuous improvement processes.

Within the parameters, *PAMid* refers to the identifier of the PAMs and *aMid* to that of the analytical microservices.



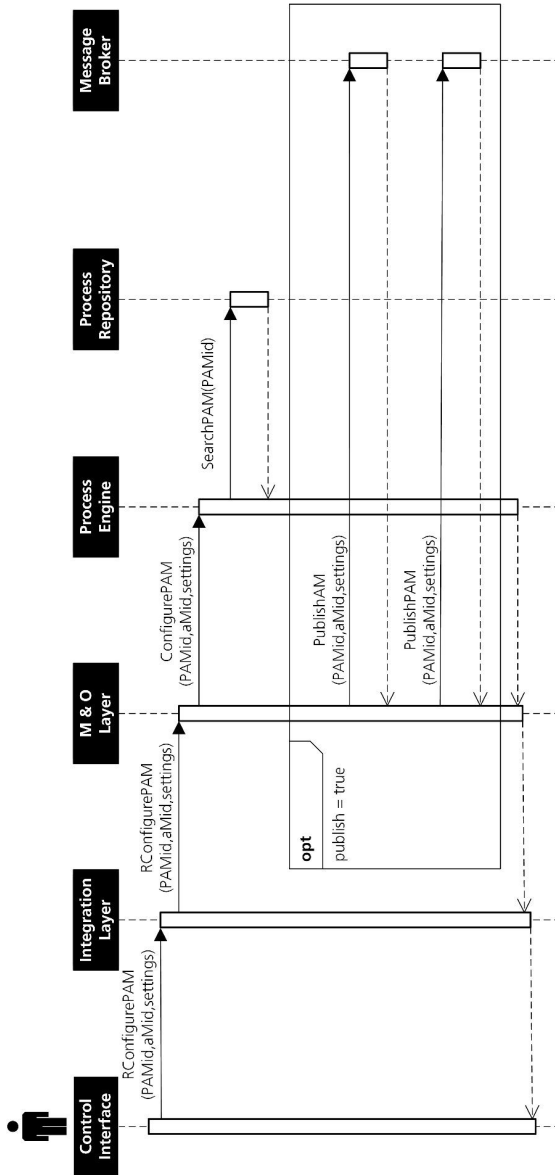


Figure 6-13: Configuration sequence

The **training sequence**, illustrated in Figure 6-14, is the obliged step before applying many – albeit not all – of the aMs.

Once the Process Engine is done selecting and instantiating the Process Coordinating Service (using the internal message *InstantiateCoordinatingService*), the latter initiates the process of training each of the affected algorithms into models (represented as a loop in the diagram). During the execution, the required data is acquired through the Integration Layer, which manages the connection – not illustrated – to the internal and external data sources.

The diagram also represents the **test sequence** as sub-process within the training – although it is possible to invoke it separately, which is not depicted in order to avoid redundancy. If the training process deems it necessary, the test sequence can be invoked in order to provide further information about the performance of a model – for example, utilizing a different data set as the one used for training.

Based on the observed performance, the Management and Operations Layer can decide if the trained model is acceptable. In the affirmative case, the model is stored. Should the model not fulfil the expected performance, the training process can be restarted with difference parameters (using the internal message *ChangeParameters*) or using a different algorithm/aM (using the internal message *ChangeaM*).

The objective of the training sequence is not only to generate models. It can also – in collaboration with the configuration sequence – contribute to changing PAMs, creating new branches, determining the best area of application, or changing the aMs utilized in order to improve the accuracy of the process.

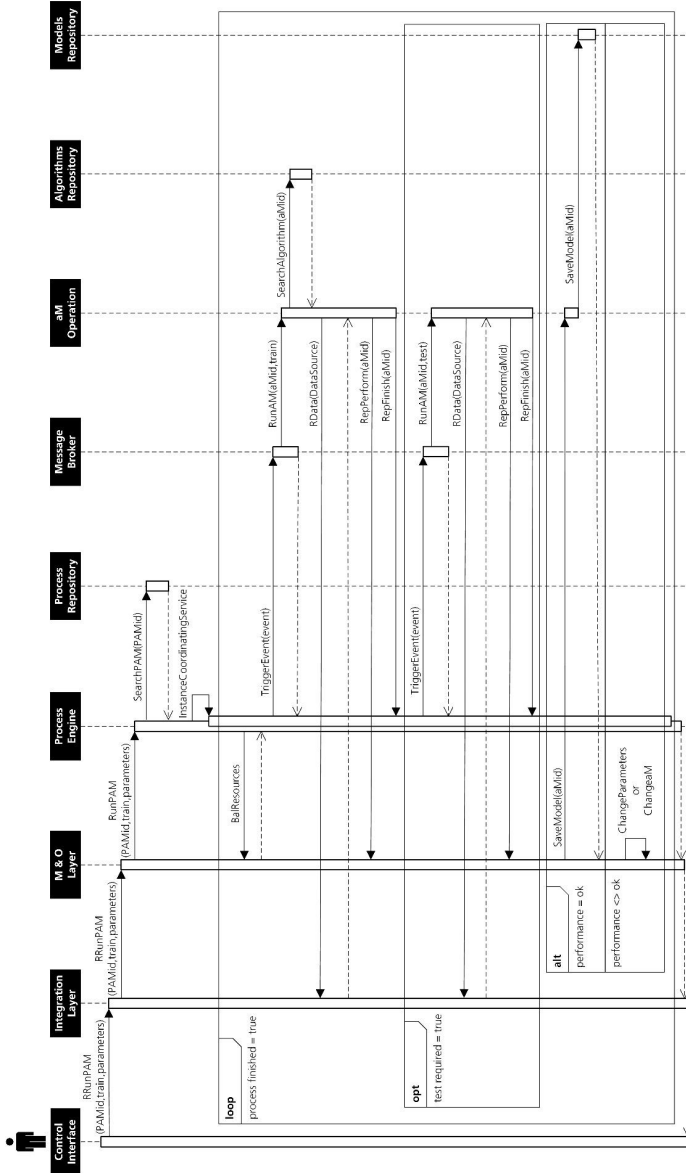


Figure 6-14: Training sequence

The messages are represented as synchronous (arrow with full line). Nevertheless, the invoking process of the PAMs and the aMs may as well function asynchronously. This can be specified by the external actor or be the result of the coordination between the Process Engine and the Management and Operations Layer in order to balance the resources.

The **application sequence**, depicted in Figure 6-15, allows the utilization of the models at the request of an external system (users or IT systems) or through internal triggering – for example, regular executions scheduled as jobs in the Management and Operations Layer.

The structure is similar to that already explained in the training sequence: a Process Coordinating Service and the involved aMs are executed until the PAM is complete. A key difference resides in the handling of the results of the analytical microservices. These are conveyed, after the execution, to the Integration Layer, which then proceeds to store them internally – for later usage or in order to be used by the following aM – and, at the finalization of the PAM, to communicate them to the calling external system.

Also of importance is the **retraining sequence**. This allows the adaptation of the PAMs and aMs to changes in the application environment. It can be triggered by a specific request or by Management and Operations Layer as part of a continuous improvement process. This could consist of regular retesting, comparisons of the results with the data about the actual occurrences in the external system (feedback), and monitoring. When the performance is detected to fall below acceptable levels, the retraining is deemed necessary.

The sequence is illustrated in Figure 6-16. It also contains an optional test sequence (in case of regular retesting) and the optional triggering of a training sequence.

As with the training sequence, the application and retraining sequences also allow the asynchronous execution of the aMs if required.

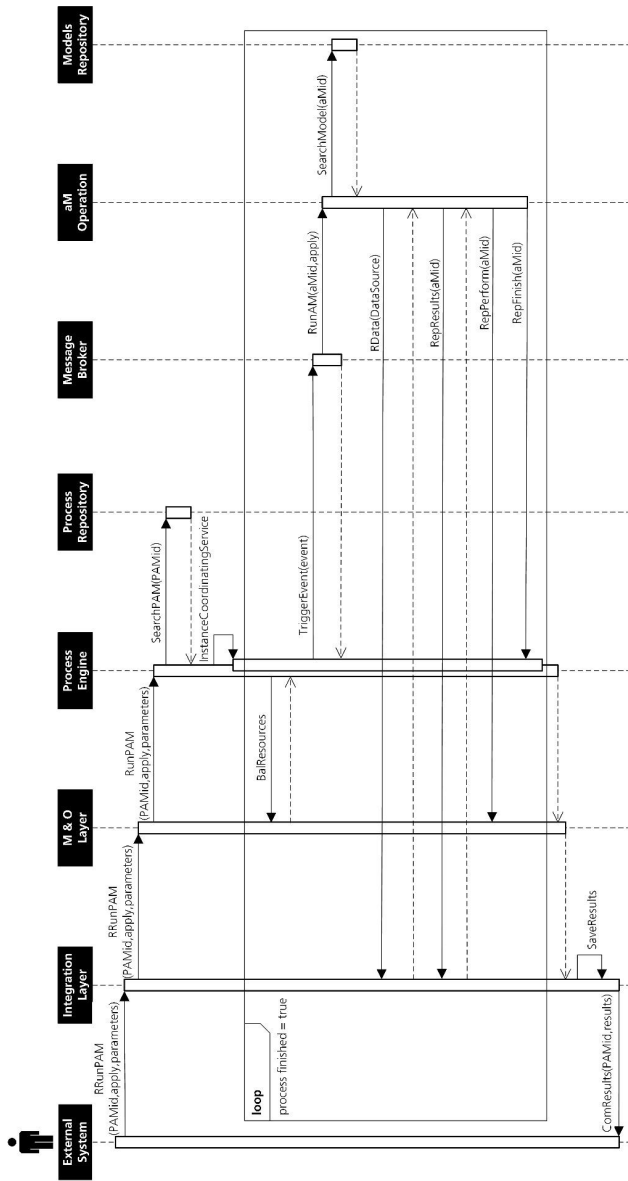


Figure 6-15: Application sequence

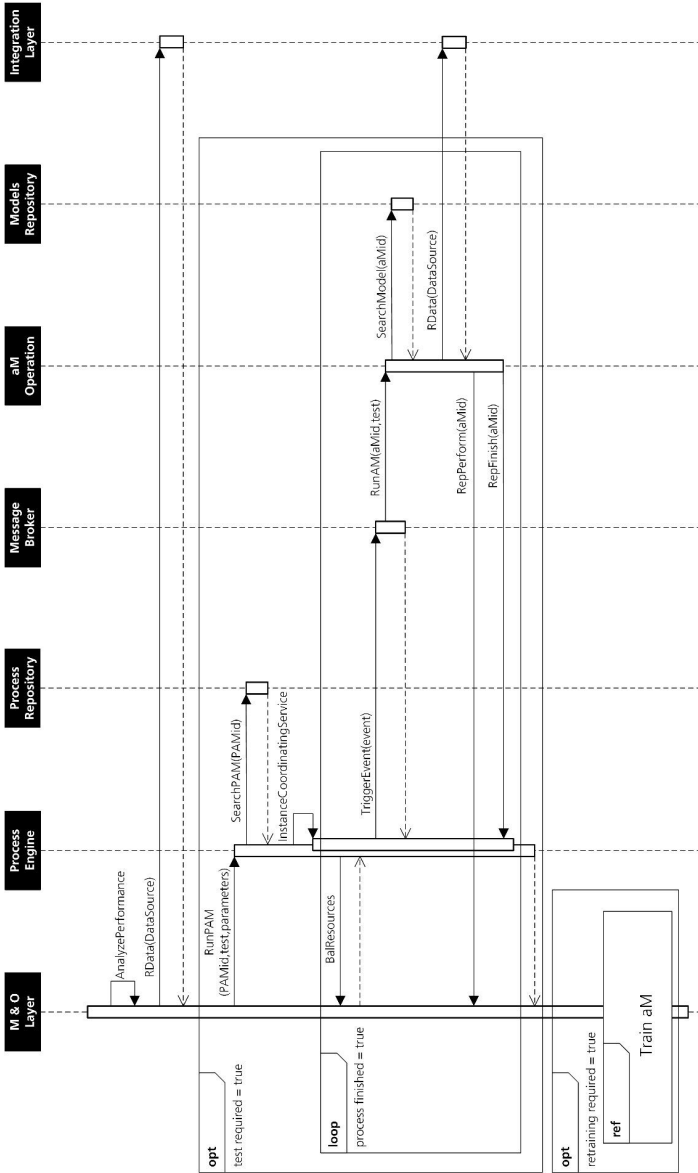


Figure 6-16: Retraining sequence

The depiction of the sequences has been simplified in order to aid their comprehensibility, showing only the main aspects. Additional actions of lesser relevance (e.g. storage of the performance data) have been omitted.

Furthermore, these sequences represent the main utilization forms of the solution. Additional utilization forms for side activities (e.g. importing an aM) are also possible.

## 6.6 Summary

The proposed solution consists of several components. The analytical microservices (aMs) encapsulate the functionalities of trainable algorithms as well as pre-trained or directly applicable models. This allows using a wide range of analytical approaches, from simple mathematical models to data mining, even facilitating the utilization of knowledge transfer. Along with enabling the bidirectional data communication, the interfaces of the aMs can receive parameters and settings as input.

Main characteristics of aMs are:

- their ability to be dynamically instantiated
- the possibility to easily select the algorithm and model best suitable for each situation
- the possibility to compose aMs into analytical processes

This allows providing analytical functionalities adapted to the needs of the underlying business processes.

The described analytical environment is composed of three layers assisting the utilization of aMs:

- The Integration Layer handles all communication within the solution and with the external environment (even covering ETL functions if needed).
- The Process Layer handles the analytical processes. It contains the Process Engine, which is responsible for managing the coordinating services controlling the execution of the corresponding analytical process and deciding which Process of Analytical

Microservices (PAM) to execute based on the available parameters. The execution of PAMs and aMs is handled using events to which they subscribe.

- The Management and Operations Layer is responsible for administrating the system, managing the execution of the functionalities and the corresponding resources. Furthermore, it monitors the performance of PAMs and aMs, triggering the required change processes (e.g. retraining of aMs) should it fall below acceptable levels. It also handles the scalability of the solution through the asynchronous execution, replication, and distribution of the aMs.

The correspondence of these components with those of the SOA Reference Architecture was analyzed in order to prove their completeness.

The analytical processes allow creating complex analytics and covering different steps of the KDD process. In the proposed solution, an aM must be contained in a process in order to be executed (even if it is only one). Analytical processes are to be invoked from the step of a business process (in an external IT system) in order to fulfill an analytical request. A Process Coordinating Service (PCS) will be instantiated in order to manage the execution of the aMs – with the corresponding analytical steps contained within each PAM – by triggering the events when needed. This combination of orchestration (performed by the PCS) and choreography (through the asynchronous communication via events) enables a loose coupling between the aMs while providing a clear overview over each process, allowing their easy management.

When selecting a PAM, it can either match the parameters of the context or just cover them (among others). The second case leads to PAMs where further decisions can be made during its execution as to which aMs should be utilized. As a result, different types of configurations of analytical processes emerge which, depending on the level of variations between the cases covered, could allow choosing between different versions of the aMs, totally different aMs, or even different process branches (which could also have a divergent number of steps). Furthermore, the parallel execution of aMs and branches within a process is also covered. The utilization of iterations within processes (e.g. for reinforcement learning) is also possible.



---

The structure of the solution can be illustrated with a component diagram, with interfaces representing the interactions between the components. Apart from those for execution and data communication, a number of additional interfaces are required, for example, for searching for PAMs, balancing the resources, publishing an aM in order to associate it to an event, etc.

Of the several possible utilization forms of the solution, five were detailed because of their importance to perform the main functionalities:

- *Configuration* allows changing the composition of PAMs and the settings.
- *Training* is used, as its name states, for training algorithms into models, as well as changing PAMs (in collaboration with configuration).
- *Testing* is used for measuring the performance of models.
- *Application* covers the utilization of a PAM (and the corresponding aMs) to fulfill a request.
- *Retraining* allows triggering the adaptation to changes in the environment.



# 7 Application Examples and Critical Evaluation

The validation of the proposed solution is challenging, as it needs to consider the current situation of the enterprise and the manufacturing environment in which it is applied, as well as how they evolve over time. Furthermore, there are crucial hidden benefits originating in the avoided losses and the efficiency gains. Likewise important is the consideration of different production contexts, as the complexity involved may influence the benefits of the solution in comparison to other alternatives. For this purpose, two validation methods are elaborated in this chapter.

The first validation procedure builds upon the evaluation of different scenarios in order to analyze a wide spectrum of situations. The assessment is based on a utility analysis of the solution alternatives in the proposed scenarios.

The second validation procedure attempts to make an economic analysis in order to complement and extend the findings of the first procedure. Additionally, data of an exemplary industrial enterprise are utilized.

## 7.1 Analysis of Application Scenarios

This section provides an evaluation of the proposed solution in different scenarios and a comparison with other approaches. The setting of the scenarios and the utilized formula are also described.

### 7.1.1 Formula for Multi Criteria Analysis

The intention is to evaluate the convenience of the proposed solution in a number of different scenarios, while analyzing at the same time the behavior in relation to influencing parameters.

For the evaluation it is necessary to recall the hypothesis of the work:

*»The construction and usage of production logistics analytics as microservices improves the former's benefit-cost ratio under the requirements of Personalized Production«*

The *benefit-cost ratio* is used to evaluate the solution and to compare it to alternative approaches: the higher the ratio, the better the approach.

With the objective of the evaluation being known (to determine the value of the benefit-cost ratio) the next step is to choose a method to perform the evaluation. As the scenarios to be proposed only intend to represent the main aspects of different situations, no deep level of detail is to be provided. This excludes the utilization of exact quantitative methods. Furthermore, the employed method should be able to consider different factors with varying importance depending on the specific context.

The decision theory provides a great of number of possibilities to support the decision making process. Based on the mentioned conditions, the evaluation method should belong to the area of Multi(ple) Criteria Decision Making (MCDM) – because of the consideration of different factors – specifically to those for Multi(ple) Attribute Decision Making (MADM), as only one main objective is to be evaluated (Götze 2008, p. 173).

Of the methods considered within MADM, the one chosen is the Utility Analysis. Zangemeister (2014, p. 45) defines this approach – known as *Nutzwertanalyse* in German – as an “analysis of a set of complex action alternatives with the purpose of ordering the elements of this set according to the decision maker's preferences regarding a multidimensional target system”. This allows considering several factors of different nature (quantitative, qualitative, assumptions, etc.) without requiring an elevated level of detail (Kühnapfel 2019, pp. 2-3). Other MADM methods, such as the Analytic Hierarchy

Process (AHP), the Multi Attribute Utility Theory (MAUT), and the Preference Ranking Organization Method for Enrichment Evaluations (PROMETHEE), require a level of detail that is not provided by the scenarios to be considered and a level of complexity unnecessary for the intended application – as such methods were developed to analyze thoroughly defined alternatives (Götze 2008, pp. 188-229).

As expressed throughout this work, the flexibility of the solution plays an important role in its utility. The evaluation considers flexibility as consisting of four elements (Slack 1983, p. 7, 1987, p. 39; Koste et al. 1999, pp. 78-79; Koste et al. 2004, p. 172):

- The **range-number** expresses “the number of possible options that a system or resource can achieve”.
- The **range-heterogeneity** “addresses the degree of difference between different options”.
- The **mobility** represents how easily it is to move from one state to another.
- The **uniformity** refers to “any alteration or deterioration of the system associated with invoking a flexible response”.

To analyze the utility of each scenario, it is necessary to create a formula considering the influencing factors (including those required for measuring flexibility). The aim is to evaluate the ability of each solution to apply analytical approaches to address the information requirements of enterprises with a specific context and needs, and the associated costs/effort.

The formula consists of the following components:

The **direct utilization (DU)** represents to which degree each solution is able to solve the analytical problems it is faced with. This will not only depend on the number of analytical approaches allowed by the solution (range-number), but also on its ability to use different techniques (range-heterogeneity) – e.g. Is it able to only use visual analytics or can more advanced data mining and machine learning approaches be utilized? An analytical solution being specifically developed for the intended purpose will also provide a higher direct utilization than the use of generic applications.

Possible values for this factor range between 1 (low ability to solve the analytical problems) and 5 (high ability to solve the analytical problems).

The **further use (FU)** represents the capability of each solution to adapt to changes in the manufacturing environment in which they are applied. As such, it covers the ease with which analytical functionalities can be update or replaced by new ones (mobility). It also considers if the adaptation causes any loss in the functionality (performance, accuracy, etc.) of the analytical functionalities (uniformity).

It is important to bear in mind that the further use will also contemplate the development efforts associated with adapting the solution. Although these could also be considered by the costs factor, this way of working avoids evaluating the same flexibility components twice. The analysis of the industrial use case (section 7.2) will, on the other hand, consider the emerging costs of the adaptations because of the characteristics of the cash flow analysis employed.

Possible values for this factor range between 1 (low mobility and uniformity) and 5 (high mobility and uniformity).

The **benefit** is composed of an addition of the direct utilization and the further use. Possible values range therefore between 0 (no benefit) and 10 (high benefit).

The **general investments (GI)** represent costs of discrete occurrence. This category covers the investment in infrastructure, the acquisition and development of the solution and analytical functionalities (with the above mentioned exception of adaptation costs), and other costs aspects related to the implementation of the solution (e.g. integration with other systems).

Possible values for this factor range between 1 (low investments required) and 5 (high investments required).

The **running costs (RC)** represent costs of continued occurrence. This category includes elements such as license costs, monthly fees, etc. Through approaches such as pay-per-use models, running costs can reduce the implementation costs – the resulting influence of time will be regarded in the cash flow analysis in section 7.2. From a flexibility

point of view, the pay-per-use models can aid the mobility of the solution. There is, however, no duplication by the evaluation of the related flexibility component, as the emerging costs are the counterpart of the mobility already considered under further use. Possible values for this factor range between 1 (low running costs) and 5 (high running costs).

The **cost** is composed of an addition of the general investments and the running costs – thus covering the main categories of IT costs (Dürr 2013, p. 98). Possible values range therefore between 0 (low costs) and 10 (high costs) – the value 0 is, however, only indicative, as it must be avoided in order to be able to calculate the ratio.

As a result, the **utility** represents the benefit-costs ratio. The higher the value, the greater the utility provided by the solution in the analyzed scenario.

Furthermore, the formula will also employ weighting factors (**w<sub>1</sub>, w<sub>2</sub>, w<sub>3</sub>, and w<sub>4</sub>**). They allow adjusting the influence of a factor depending on the context, with values ranging between 0 (no importance) and 1 (full consideration). They are necessary in order to better represent the reality of each enterprise type. For example, should an enterprise consider the burden (costs) of implementing an analytical solution more important than the benefit of being able to adapt the analytical functionalities (a usual situation in small enterprises), the weight of the further use factor will be below 1.

The resulting formula for the assessment of the benefit-cost ratio – which will be referred to as **utility** – is shown in Equation 1.

$$\textit{Utility: } \frac{\textit{Benefit}}{\textit{Cost}} = \frac{w_1 \times DU + w_2 \times FU}{w_3 \times GI + w_4 \times RC} \quad (1)$$

### 7.1.2 Setting of the Scenarios

The objective of the scenarios is to analyze how the utility of the proposed solution behaves under different conditions and how this behavior compares to those of the alternative approaches mentioned in section 5.4. The scenarios will therefore be

composed of the intersection of two determining elements: the enterprise type and the considered function group.

The enterprise types (E) describe organizations with products and manufacturing systems of different complexity. The main condition is that a degree of personalization of the products must be possible.

As stated in section 2.1.2, the complexity in an enterprise consists of four dimensions: variety, heterogeneity, dynamic, and opacity. To establish the relation between each enterprise type and their corresponding complexity according to these dimensions, it is first necessary to characterize them. For these purpose, the following parameters are to be used:

- The intricacy of their products (**product intricacy**). This depends on factors such as the number of elements in each product and how intricate their composition is. This parameter is equivalent to the product complexity mentioned in section 2.1.2 (the term *intricacy* is used to avoid confusions with the complexity of the enterprise types).
- The number of basic product types offered (**basic types**).
- The number of possible product variants offered (**variants**). These are a result of configuring/personalizing the basic types (being influenced by the product intricacy).
- The number of different raw material employed (**raw materials**). This parameter is also directly influenced by the product intricacy.
- The number of work stations necessary to manufacture the products (**work stations**).
- The typical yearly sales of an enterprise within the enterprise type (**yearly sales**), expressed in euros.

The variety dimension is influenced by the number of basic product types offered. The heterogeneity dimension contemplates the number of possible variants.

The dynamic dimension is determined by how sensitive the enterprise type is to changing conditions in:

- The source market, influenced by the number of raw materials. Many raw materials mean a high dependency on different vendors.



- The manufacturing environment, influenced by the number of work stations. It will reflect, for example, the possible effects of a change in the manufacturing technology.
- The sales market, influenced by the number of possible variants. The more customer requirements addressed, the more sensitive the enterprise becomes to changes in the market.

Lastly, the opacity dimension will consider the product intricacy as well as the number of work stations, raw materials and variants. The more elements involved and the more intricate they are integrated with each other, the higher the possibility that the enterprise loses the capability to understand the situations in the supply chain and in the own manufacturing processes.

The yearly sales are used mainly to provide an idea of the size of a typical example enterprise within the enterprise type (which is also reflected in the number of work stations).

#### *Enterprise types and Function Groups*

The three enterprise types to be considered are:

- **E1** represents a company with relatively simple products and reduced personalization possibilities (e.g. electronic control units where the adaptation to the main product types is done through changes to the embedded software). This type can be exemplified by an enterprise characterized by the following parameters:
  - product intricacy: low
  - basic types: 10
  - variants: 1,000+
  - raw materials: 100
  - work stations: 20
  - yearly sales: less than 100 million euro

Enterprises in this type will possess a low heterogeneity and a low to medium variety. A high variety would be possible, it would however imply diversifying the business. They will also typically have a low dynamic (because of their low exposure) and a low

to medium opacity (which could be high if, for example, they have little visibility over their supply chain or processes).

- **E2** represents providers of consumer products with several degrees of personalization (e.g. the provision of furniture and construction elements such as windows, which allow configuring the size, material, frame, etc.). This type can be exemplified by an enterprise characterized by the following parameters:
  - product intricacy: medium
  - basic types: 70
  - variants: 50,000+
  - raw materials: 350
  - work stations: 50
  - yearly sales: between 100 and 1,000 million euro

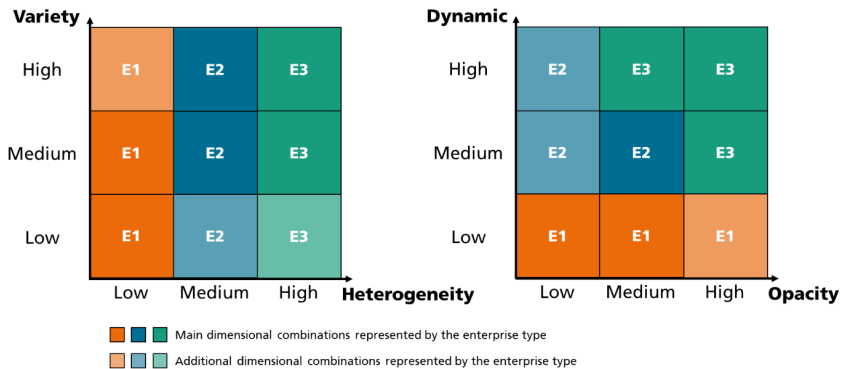
Enterprises in this type will possess a medium heterogeneity and a medium to high variety. A low variety would also be possible, it would however represent small producing enterprises with a limited number a products. They will typically have a medium opacity (which could be low with a good control over the supply chain and manufacturing processes) and a medium dynamic (which could be high if they are in a rapidly changing environment).

- **E3** represents providers of industrial products. They normally offer a limited number of basic types which are highly personalizable (e.g. machine tools). This type can be exemplified by an enterprise characterized by the following parameters:
  - product intricacy: high
  - basic types: 35
  - variants: 100,000+
  - raw materials: 1,500
  - work stations: 145
  - yearly sales: greater than 1,000 million euro

Enterprises in this type will possess a high heterogeneity and a medium to high variety. A low variety would also be possible, it would however represent the production of a limited number of highly personalizable products. Giving the high number of elements

and the corresponding exposure, both the opacity and the dynamic will be medium to high.

The enterprise types and parameters were modelled after real enterprises. Their relations to the complexity dimensions are illustrated in Figure 7-1.



**Figure 7-1: Relations between complexity dimensions and enterprise types**

The function groups (F) refer to different production management functions that need to be supported by the analytical solution. These are based on the PPC function assignment presented in Figure 2-4 and are as follows:

- **F1:** Program planning
- **F2:** Material requirements planning + F1
- **F3:** Capacity requirements planning + F2
- **F4:** Production control + F3

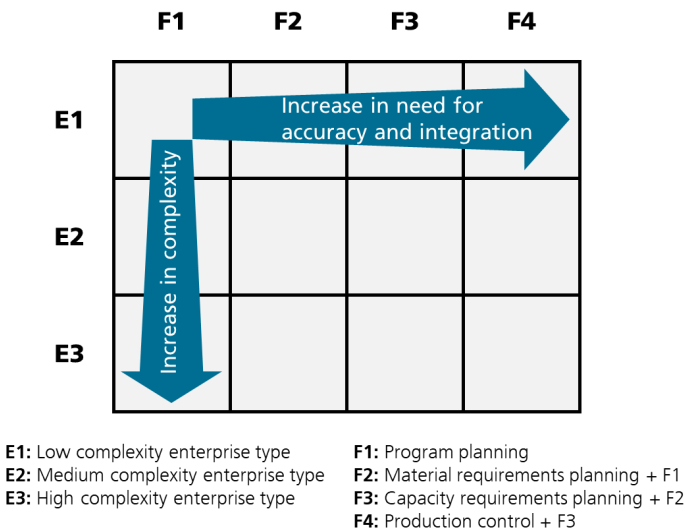
Furthermore, the functions within the function groups are additive. This means that every function group considers not only the corresponding PPC functions – as illustrated in Figure 2-4 – but also the ones of the previous function groups (in the above presented order). For example, F2 will include, in addition to the functions for material requirements planning, those for program planning in F1.

This supports simplicity, as the main objective is to analyze the increment in the number and intricacy of functions.

### *Quantification of Scenarios*

Before performing the evaluation of the scenarios, it is necessary to take several conditions into account:

- The considered scenarios are subject to the challenges presented by Personalized Production, originating – as stated in chapter 2 – in the deliver phase (e.g. expectancy of delivery times close to mass production) and the disposition phase (e.g. increasingly turbulent production environment, unknown demand of raw and semi-finished goods, etc.).
- The assignment of values to the factors for each approach is not absolute but performed in a relative manner to the valuation of the other approaches.



**Figure 7-2: Requirements of scenarios**

The scenarios are to be quantified using the components of the utility evaluation formula taking into account the attributes of each enterprise type, the characteristics of each function group, and the above explained conditions. In Figure 7-2 it is described how a scenario is created by combining an enterprise type and a function group. Furthermore, it shows how the main characteristics of the scenarios change with an increase in complexity from top to bottom (mainly driven by the enterprise type) and a rise in the need for accuracy and integration from left to right (stipulated by the needs of the function groups, which become increasingly intricate and sophisticated).

**Table 7-1: Quantification of scenario E1 – F1**

Scenario			Benefit				Cost				Utility
			W <sub>1</sub>	DU	W <sub>2</sub>	FU	W <sub>3</sub>	GI	W <sub>4</sub>	RC	
E1	F1	Business Intelligence Solutions	0.7	4	0.5	2	1	2	1	1	1.27
	F1	Data Mining Suites	0.7	4	0.5	4	1	3	1	1	1.20
	F1	Tailored Solutions	0.7	5	0.5	1	1	2	1	1	1.33
	F1	Analytics as a Service	0.7	4	0.5	4	1	2	1	2	1.20
	F1	Analytical Microservices	0.7	5	0.5	5	1	3	1	2	1.20

DU: Direct Utilization; FU: Further Use; GI: General Investments; RC: Running Costs; W<sub>1</sub>: Weight Direct Utilization; W<sub>2</sub>: Weight Further Use; W<sub>3</sub>: Weight General Investments; W<sub>4</sub>: Weight Running Costs

As an example, in Table 7-1 the quantification of the scenario E1 – F1 is shown. The combination of the simplest enterprise type with the least intricate function group allows for a high direct utilization of all solutions, although this is slightly better in the cases of the tailored solutions and the analytical microservices, as they have the highest accuracy. The further use of the business intelligence solutions and the tailored solutions is impaired by the high efforts necessary to adapt the solutions to changes; the analytical microservices, on the other hand, enjoy a good rating thanks to their high flexibility. The weight affecting the direct utilization (W<sub>1</sub>) has a value lower than 1 to consider the fact that enterprises of type E1, which usually are small, place more importance on cost factors than on benefit. This weight can, however, increase in the more sophisticated function groups, as a low accuracy of the solution could have a more detrimental effect. The weight

affecting further use ( $W_2$ ) is also lower than 1 because of the low dynamic of E1 enterprises.

Intricate solutions – data mining suites and analytical microservices – require higher general investments than their counterparts, though the final rating will also depend on the enterprise complexity (e.g. the bigger the enterprise, the higher the implementation costs). Highly flexible software solutions – Analytics as a Service and analytical microservices – will probably also have higher running costs, especially as they are prone to be used within pay-per-use models.

The example in Table 7-2 shows the quantification of scenario E2 – F2. The direct utilization of solutions that do not allow for a specific adaptation to the production requirements – a fact that becomes more important with increasing complexity and sophistication of enterprise type and function group – is reduced in comparison to the previous scenario. The further use of solutions less flexible than the analytical microservices is also reduced due to the higher dynamic of E2 enterprises (with more materials and variants). Accordingly,  $W_1$  and  $W_2$  are also adjusted upwards.

**Table 7-2: Quantification of scenario E2 – F2**

Scenario	Solution	Benefit				Cost				Utility	
		$W_1$	DU	$W_2$	FU	$W_3$	GI	$W_4$	RC		
E2	F2	Business Intelligence Solutions	0.8	2	0.8	1	1	3	1	2	0.48
	F2	Data Mining Suites	0.8	3	0.8	3	1	3	1	2	0.96
	F2	Tailored Solutions	0.8	5	0.8	1	1	4	1	1	0.96
	F2	Analytics as a Service	0.8	3	0.8	3	1	2	1	2	1.20
	F2	Analytical Microservices	0.8	5	0.8	5	1	3	1	3	1.33

DU: Direct Utilization; FU: Further Use; GI: General Investments; RC: Running Costs;  $W_1$ : Weight Direct Utilization;  $W_2$ : Weight Further Use;  $W_3$ : Weight General Investments;  $W_4$ : Weight Running Costs

Due to the higher analytical requirements of the function groups, the general investments needed by the tailored solutions increases, representing the development effort involved. Likewise, the costs of business intelligence solutions and data mining suites increase in

order to represent the need for solutions with more features (and therefore more expensive).

The example in Table 7-3 shows the quantification for the same enterprise type as before (E2) but with a more sophisticated function group (F3). One resulting effect is that  $W_1$  increases its value to 1. This is due, as explained before, to the higher importance of accuracy in more sophisticated function groups. Also, demand-focused function groups such as F1 and F2 could profit in E2 from Assemble-to-Order approaches, allowing to pre-plan several modules and components, reducing the stress on the demand forecast functions. F3 and F4 are, on the other hand, more affected by the intricacy of the production (more raw materials and work stations), hence increasing the need for accuracy.

**Table 7-3: Quantification of scenario E2 – F3**

Scenario	Solution	Benefit				Cost				Utility	
		$W_1$	DU	$W_2$	FU	$W_3$	GI	$W_4$	RC		
E2	F3	Business Intelligence Solutions	1	2	0.8	1	1	3	1	2	0.56
	F3	Data Mining Suites	1	3	0.8	2	1	3	1	2	0.92
	F3	Tailored Solutions	1	5	0.8	1	1	4	1	1	1.16
	F3	Analytics as a Service	1	3	0.8	3	1	3	1	2	1.08
	F3	Analytical Microservices	1	5	0.8	5	1	3	1	3	1.5

DU: Direct Utilization; FU: Further Use; GI: General Investments; RC: Running Costs;  $W_1$ : Weight Direct Utilization;  $W_2$ : Weight Further Use;  $W_3$ : Weight General Investments;  $W_4$ : Weight Running Costs

Another effect is the increase in general investments for Analytics as a Service. This is due to the additional effort required to integrate the solution in the manufacturing environment – a requirement that increases in manufacturing-based function groups (F3 and F4).

The quantification of scenario E3 – F4 is shown in Table 7-4. As to be expected, less capable and less flexible solutions see their direct utilization and further use reduced. It must also be mentioned that, in order to balance the inadequacy of an approach (e.g.

business intelligence solutions for functions in F4), the cost factor is increased. This represents, for example, additional development efforts. However, it does not guarantee a good benefit (direct utilization and further use) but rather the provision of minimal required usability, as the solution will still be technologically constrained.

**Table 7-4: Quantification of scenario E3 – F4**

Scenario		Solution	Benefit				Cost				Utility
			W <sub>1</sub>	DU	W <sub>2</sub>	FU	W <sub>3</sub>	GI	W <sub>4</sub>	RC	
E3	F4	Business Intelligence Solutions	1	1	1	1	1	5	1	2	0.29
	F4	Data Mining Suites	1	2	1	2	1	4	1	2	0.67
	F4	Tailored Solutions	1	5	1	1	1	5	1	1	1.00
	F4	Analytics as a Service	1	2	1	2	1	3	1	2	0.80
	F4	Analytical Microservices	1	5	1	5	1	3	1	3	1.67

DU: Direct Utilization; FU: Further Use; GI: General Investments; RC: Running Costs; W<sub>1</sub>: Weight Direct Utilization; W<sub>2</sub>: Weight Further Use; W<sub>3</sub>: Weight General Investments; W<sub>4</sub>: Weight Running Costs

### 7.1.3 Evaluation of Scenarios

The results of the evaluation of the proposed analytical microservices solution in the twelve scenarios created by combining enterprise types and function groups are shown in Table 7-5.

While the production management functions considered in function groups F1 and F2 are mainly affected by the material intricacy (product intricacy and raw materials), function groups F3 and F4 are more sensitive to structural intricacy. The latter is not only dependent on the number of work stations but also on product intricacy, as this will influence the configuration of the associated manufacturing processes.

Two clear patterns can be recognized in Table 7-5:

1. The more complex the production of the enterprise type, the greater the utility of the proposed solution



2. The more production management functions supported by the proposed solution and the more sophisticated their nature, the greater its utility

Complex production environments (e.g. materials with many personalizable features) and demanding production management systems (e.g. requiring predictive capabilities for a large number of different materials) increase the necessity for more and accurate analytical algorithms and models – in many cases, for different versions of the same analytical approach.

**Table 7-5: Utility of the microservices-based solution in the evaluated scenarios**

Enterprise Type		Function Group			
		F1	F2	F3	F4
		Program Planning	Material Requirements Planning	Capacity Requirements Planning	Production Control
E1	<b>Product intricacy:</b> low <b>Basic types:</b> 10 <b>Variants:</b> 1,000+ <b>Raw materials:</b> 100 <b>Work stations:</b> 20 <b>Yearly sales:</b> LT 100 MM € <b>Variety:</b> low to medium <b>Heterogeneity:</b> low <b>Dynamic:</b> low <b>Opacity:</b> low to medium	1.20	1.20	1.30	1.30
E2	<b>Product intricacy:</b> medium <b>Basic types:</b> 70 <b>Variants:</b> 50,000+ <b>Raw materials:</b> 350 <b>Work stations:</b> 50 <b>Yearly sales:</b> BT 100 & 1,000 MM € <b>Variety:</b> medium to high <b>Heterogeneity:</b> medium <b>Dynamic:</b> medium <b>Opacity:</b> medium	1.33	1.33	1.50	1.50
E3	<b>Product intricacy:</b> high <b>Basic types:</b> 35 <b>Variants:</b> 100,000+ <b>Raw materials:</b> 1500 <b>Work stations:</b> 145 <b>Yearly sales:</b> GT 1,000 MM € <b>Variety:</b> medium to high <b>Heterogeneity:</b> high <b>Dynamic:</b> medium to high <b>Opacity:</b> medium to high	1.67	1.67	1.67	1.67

This also means that the inverse proposition has to be examined: for environments of low complexity and requirements, the utility of the solution may be lower than that of other approaches. The evaluation of the alternative approaches described in section 5.4 is presented in Table 7-6 in order to perform this comparison (a table with a detailed valuation of factors and weights is given in Appendix B).

**Table 7-6: Utility comparison of the different approaches**

Scenario		Solution				
		Business Intelligence Solutions	Data Mining Suites	Tailored Solutions	Analytics as a Service	Analytical Microservices
E1	F1	1.27	1.20	1.33	1.20	1.20
	F2	1.03	1.20	1.33	1.20	1.20
	F3	0.65	0.98	1.13	1.10	1.30
	F4	0.65	0.98	0.90	1.10	1.30
E2	F1	0.80	1.28	1.20	1.60	1.33
	F2	0.48	0.96	0.96	1.20	1.33
	F3	0.56	0.92	1.16	1.08	1.50
	F4	0.30	0.92	1.16	0.92	1.50
E3	F1	0.80	1.40	1.20	1.40	1.67
	F2	0.60	1.20	1.20	1.20	1.67
	F3	0.29	0.67	1.00	0.80	1.67
	F4	0.29	0.67	1.00	0.80	1.67

As stated when comparing the approaches, they differ in the range of analytical capabilities offered. This is especially true for the business intelligence solutions, which have a reduced area of application, and to some degree for the data mining suites and Analytics as a Service. For instance, the business intelligence solution in scenario E3 – F3 becomes overly expensive when trying to address the complexity requirements and even then it is only able to deliver a low benefit (as explained in the section on quantification of scenarios).

Analytics as a Service can be considered as similar to data mining suites, with enhanced modularity. A disadvantage presented by these approaches – apart from the above mentioned limitation in their capabilities – is the lack of integration with the IT landscape,

reducing their benefit in function groups depending on the creation of feedback loops (F3 and F4).

Tailored solutions present the best utility for both, simple and very complex situations. They offer a specifically created, adapted, and integrated solution, though not without drawbacks. Their main disadvantages are the high development costs (which increase with the amount of algorithms and models required) and their inability to flexibly adapt, requiring new development efforts (reflected in the low further use).

The utilization of tailored solutions in simple scenarios is favored by the fact that little development effort is required. In very complex scenarios, on the other hand, the benefit of tailored solutions manages to outweigh the costs. These solutions are, however, outperformed by the flexibility of analytical microservices. In scenarios in-between, a relatively inexpensive solution with some modularity (e.g. data mining solutions) can achieve a greater utility than tailored solutions.

As conjectured, the utility of analytical microservices exceeds that of other solutions in complex situations, but falls short in scenarios with low requirements. Analytical microservices profit mainly from their flexibility and their capability to provide accurate solutions in each situation. Their cost structure is relatively good, although in many cases worse than that of other approaches. However, competing approaches will also see their costs increase in order to address complex situations. Especially important are the running costs, as analytical microservices would probably be marketed on a pay-per-use model. These are thus valued higher than in the other approaches in order to represent the additional load.

## 7.2 Analysis of an Industrial Use Case

Within this section, an industrial use case will be described and the application of the proposed solution to it will be evaluated. Additionally, a comparison with the tailored solutions approach will be performed.

### 7.2.1 Description of the Industrial Use Case

To further validate the results of the scenario evaluation, this section will focus on the analysis of an industrial use case.

The chosen industrial use case represents a manufacturer of consumers electronics located in Germany. The data was acquired as part of a research project with a focus on the development of a software prototype based on the approach presented in this work.

The analyzed production site concentrates on the fabrication and commercialization of around 80 basic product types, composed by workstations, servers, thin clients, and storage systems. The production has a combined daily capacity of 13,000 units (12,000 workstations and thin clients, 1,000 servers and storage systems) and requires the utilization of several hundred types of raw materials.

Theoretically, the production site should be able to manufacture more than 900,000 product variants. This variety, however, is highly influenced by two main features with a great number of options: the processor (up to 20 alternatives) and the hard disc drive (more than 100 alternatives).

The use case concentrates on the manufacturing of motherboards to be later employed in other products. The production lines combine SMD (surface-mounted device) and THT (through-hole technology) processes in about 110 work stations (distributed among 10 lines). Approximately 340 different types of motherboards are produced (with continuous changes). An analysis of the data showed that, in average, an annual 23% of the motherboard types to enter production have never been produced before and that 15% are new to the line they are being produced in.

The analytical approach is intended to support production planning by predicting the processing times of the different motherboard types.

Comparing the use case to the scenarios analyzed in the previous section, the most similar enterprise type would be E2, particularly in regard to product and manufacturing intricacy. This requires to some extent to disregard the features causing the elevated number of

product variants. This is possible given that these features (the processors and the hard disc drives) are not directly produced by the enterprise and are just added during final assembly. Furthermore, the different lines share the same models of several machines.

The intended utilization – planning optimization through predictive capabilities – can be considered as concerning the functions within group F3 (capacity requirements planning).

As part of the already mentioned research project, the business models possibly used for the approach were discussed with the partners, integrated by providers of production software and data analytics. It was decided to focus on a pay-per-use business model for the analytical microservices – paying a monthly fee for each algorithm and basic model – and on a subscription-based business model for the corresponding platform – paying a monthly fee that additionally covers services such as updates and support. Each algorithm and basic model can be instantiated as often as desired (e.g. training a different model for each material type) without having to pay an additional fee. The business model also allows for charging a one-time price for the microservices if the user wants to purchase them.

## 7.2.2 Evaluation of the Industrial Use Case

The chosen evaluation methodology is a cash flow analysis, which provides a quantitative comparison of the costs behavior of the considered approaches. A reduction in the cost factor by the proposed approach will – provided the benefit is at least the same as in the competing approach – validate the hypothesis.

The analytical microservices will be compared with the tailored solutions. This has two reasons:

- The tailored solutions delivered the best results (except for the analytical microservices) in scenario E2 – F3, which is comparable to the current use case.
- The approach considered by the enterprise at the time of the research project was a tailored solution.

The parameters used for the cash flow analysis of the use case are shown in Table 7-7. The monetary values – rates, efforts, and monthly fee – originate from the discussion with the mentioned research project partners and an investigation of the data analytics market.

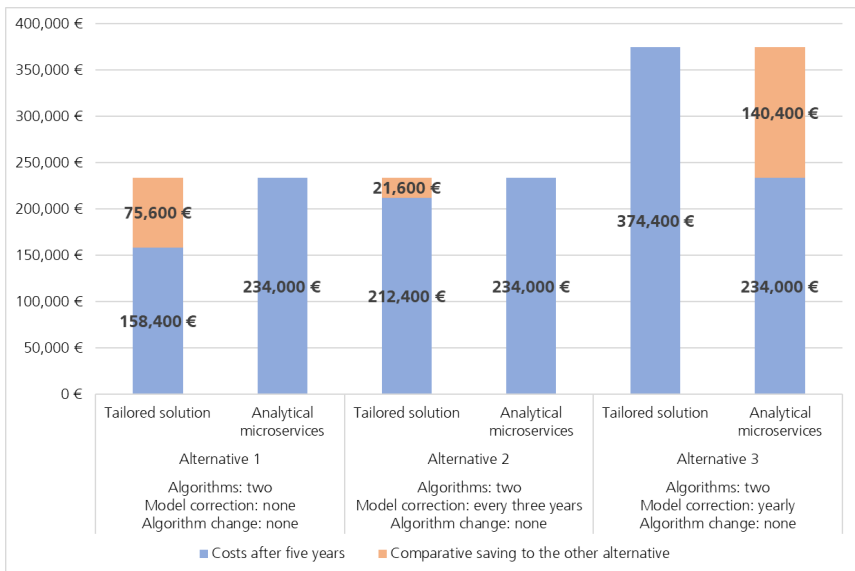
**Table 7-7: Parameters of the industrial use case**

	<b>Parameter</b>	<b>Value</b>
	Daily rate	1,500 €/person-day
<b>Tailored Solution</b>	Basic development project (base cost)	16,500 € – 11 person days
	Monthly fee (platform, licenses, etc.)	1,500 €
	Factor of base cost for an additional algorithm	60%
	Effort for addition of a model	4 person days
<b>Analytical Microservices</b>	Implementation costs	6,000 € – 4 person days
	Monthly fee – platform	800 €
	Monthly fee – analytical microservice	1,500 €

Additionally, the use case analysis assumes that eight analytical models are required in order to cover the specific behavior of each material type and line – a plausible number based on an evaluation of the manufacturing data (e.g. through clustering to visualize the relations between attributes), the amount of material types (motherboards), the number of lines (with dissimilarities between them), and the manufacturing dynamic (introduction of new materials or assignment to another line). It is worth mentioning that every time an algorithm is generated, the creation of one model is already considered in the effort. Only the remaining seven models will then represent an additional effort.

As addressed in previous chapters, the analytical microservices could offer several advantages in the support of the data preparation process. However, the corresponding costs in the use case analysis are considered, for the purpose of simplification, to be equal in both approaches.

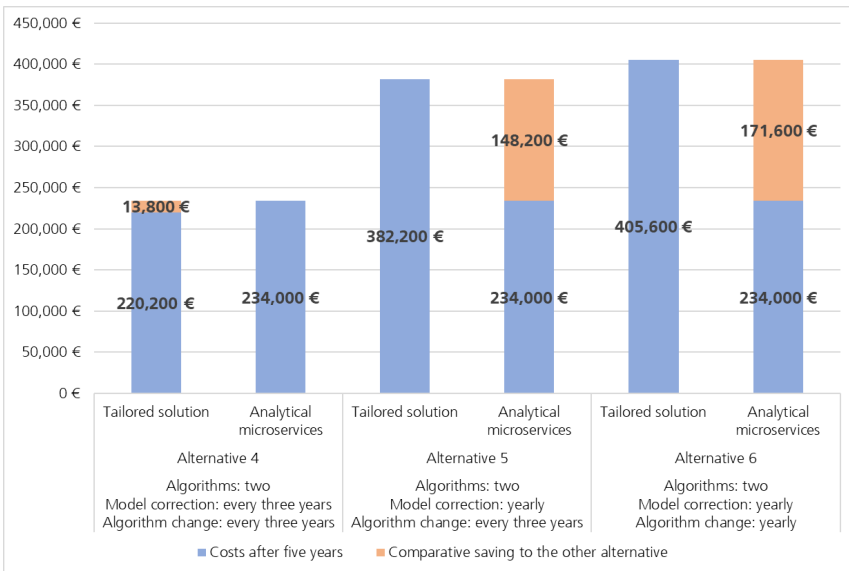
In Figure 7-3 a comparison between the analysis alternatives 1 to 3 is shown. In each alternative, the term *algorithms* refers to the number of different algorithms employed – paying, in the case of analytical microservices, a monthly fee for each one of them. The term *model correction* refers to re-generation of the models in order to adapt to the changing manufacturing environment (caused, for example, by the influx of new materials). The term *algorithm change* refers to the utilization – including selection and, in the case of tailored solutions, development – of a new algorithm in order to adapt to changes. The analyzed cash flow corresponds to a five years period.



**Figure 7-3: Comparison of cash flows of alternatives 1 to 3**

The evaluation of alternatives 1 to 3 seems to confirm and extend the findings of the scenario evaluation. In alternative 3, with its high dynamic characterized by constant model correction, analytical microservices are able to take advantage of the burden bore by the tailored solutions because of the continuously incurring development costs. In contrast, the flexibility of the analytical microservices allows an almost seamless

adaptation without a considerable extra effort. On the other hand, static alternatives, like 1 and 2, with infrequent to no model corrections, present the best conditions for tailored solutions. In these cases, the constantly incurring monthly fee becomes a burden for analytical microservices. However, the first two alternatives will probably not meet the criteria of Personalized Production, as such static situations are hardly possible.

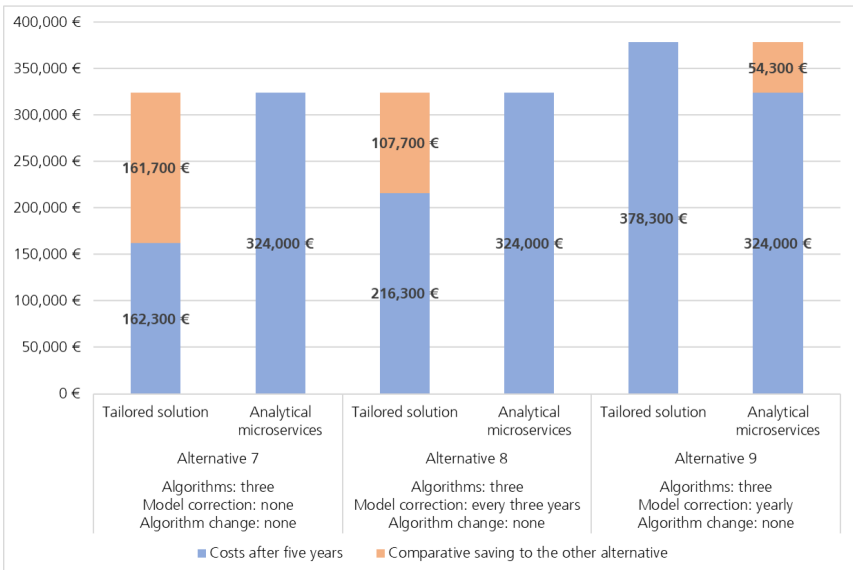


**Figure 7-4: Comparison of cash flows of alternatives 4 to 6**

Alternatives 4 to 6 – shown in Figure 7-4 – add more dynamic to the previous alternatives through the introduction of an algorithm change. The comparative position of analytical microservices is improved by this factor, being slightly disadvantageous only in alternative 4. The repeatedly incurring costs of the new algorithm burden the tailored solution. Analytical microservices, on the other hand, benefit from their interchangeability: once an algorithm is replaced, it is no longer necessary to pay the monthly fee for it. It is important to consider that this advantage may be reduced when compared to other solutions using a pay-per-use model (e.g. Analytics as a Service).

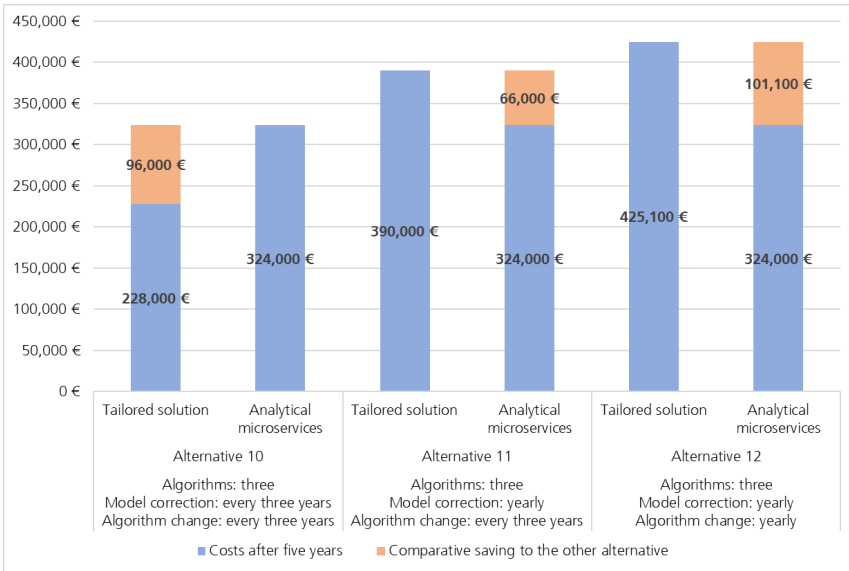


Alternative 6 is characterized by a great dynamic, with yearly changes in models and algorithms. Thus, it might be regarded as the one to better represent the conditions of Personalized Production. It is also the alternative which shows the biggest financial advantage of analytical microservices.



**Figure 7-5: Comparison of cash flows of alternatives 7 to 9**

Alternatives 7 to 12 – shown in Figure 7-5 and Figure 7-6 – furnish the already analyzed alternatives with an additional algorithm (three instead of two). Being burdened by the monthly fee of the extra algorithm, the cost of the analytical microservices increase more than those of the tailored solutions. This reduces the advantage of analytical microservices, worsening their comparative position in the more static alternatives. The cost difference in alternative 7, for example, is greater to the one in alternative 1 (the corresponding alternative with two algorithms). Analytical microservices remain the best option for most dynamic alternatives (e.g. alternatives 9 and 12), being favored by the need for continuous changes.



**Figure 7-6: Comparison of cash flows of alternatives 10 to 12**

The advantage of analytical microservices can be increased by utilizing them in a more complex environment. It is even conceivable that such a situation would accompany the necessity of a new algorithm.

One way to represent the increase in complexity is the need for additional models. The behavior of the use case under such conditions is shown in Appendix C.

Furthermore, it would also be possible to reduce the negative effects of the monthly fees for analytical microservices by making a purchase with a one-time payment. As stated before, such a methodology was not ruled out by the research project partners. The expected price would, however, represent between three and five years of utilization with the pay-per-use model. The downside of this method would, therefore, be that the utilization of the analytical function should be foreseeable in the long term. This would be possible for some special cases – e.g. an analytical microservice for a machine with a lifespan of several years to a decade.

As stated at the beginning, the cash flow analysis concentrates on costs, assuming that the benefit remains constant. One way in which the situation in reality may differ is through the consideration of the adaptation time. While analytical microservices were conceived to rapidly adapt to changes in the manufacturing environment – even being able to detect the need to do so – tailored solutions would require considerably longer. This may generate several losses, for example, because of lack of efficiency or wrongly predicted procurement and delivery times.

### 7.3 Critical Evaluation

The adequacy of an analytical microservices solution to address problems in production logistics, in particular in the context of Personalized Production, can be assessed from different points of view.

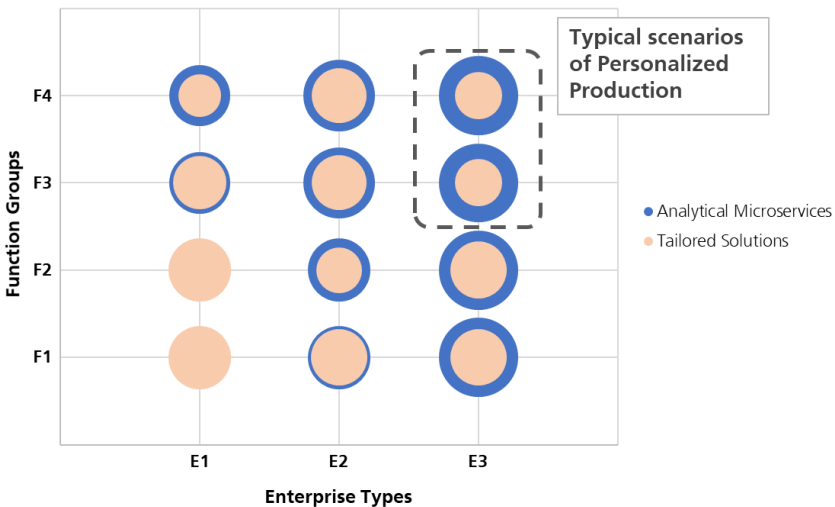
From the perspective of production logistics, analytical approaches provide a way for software-based functions to deal with increasingly difficult issues containing a great number of influencing factors and unknown elements. It should, however, be considered that not always highly accurate analytical solutions are needed but rather those that fit the requirements of an enterprise and its manufacturing context. This allows savings in time, money, and effort. Regarding the application area proposed for the solution, two factors need to be taken into account:

- Personalized Production, combining an increasing complexity with the requirements of mass production, will require some level of data analytics-based approaches in order to fulfil the logistic objectives.
- The dynamic character of Personalized Production makes the utilization of an adaptable solution a necessity. Furthermore, a flexible solution is necessary in order to address heterogeneity, enabling the provision of specific approaches for each situation. Analytical microservices fulfil both of these requirements.

From a technical perspective, the main asset of the solution is its flexibility, which allows not only to adapt to the requirements of the manufacturing environment but also to those

of the analytical approaches. This also has a disadvantage, as flexible software approaches like microservices-based solutions can become very complex, affecting their usability. In order to create a solution that can be used in the real world, the proposed concept relinquishes a certain amount of flexibility by using a hybrid approach combining microservices and SOA. However, through the correct utilization and administration of Process Coordinating Services, there are no hurdles for the constitution of flexible analytical processes. The result is the creation of a solution that is easy to comprehend, implement, and maintain.

Nevertheless, the constitution of a flexible solution still has a drawback: its cost structure. This is clearly reflected in the third perspective: the utility.



**Figure 7-7: Utility comparison of analytical microservices and tailored solutions**

The utilities of analytical microservices and of tailored solutions (represented by the diameter of overlapping bubbles) in different scenarios are depicted in Figure 7-7. The tailored solutions were chosen for comparison as they provide the same capability to

develop highly accurate analytical approaches. It becomes clear that the utility of analytical microservices is different from that of tailored solutions in enterprise types with high complexity and a high number of variants. Furthermore, the more sophisticated the function group, the greater the advantage of the analytical microservices solution. These are cases where the flexibility of analytical microservices gives them an edge. As scenarios such as E3 – F3 and E3 – F4 can be considered as representative for Personalized Production, the convenience of the proposed solution in this manufacturing context is visible.

On the other hand, the opposite argument should also be taken into account. As the reviewed trend in utility is replicated when comparing analytical microservices against other approaches, this means that the proposed solution is inadequate for application in simple scenarios. In such cases, the capabilities and flexibility of the approach are excessive and the resulting costs become a burden, in particular when using pay-per-use models.

When considering the economic benefit, two additional aspects are worth considering:

- In some situations, the possession of extra flexibility makes sense for enterprises, even at the expense of higher costs. It is, for example, the case of organizations that have a clear development roadmap according to which flexibility will be required in the future. In this scenario, the savings of using a cheaper system at the beginning will be offset by the costs of implementing a new one.
- The examples presented in this work are based on simplifications. For instance, data preparation costs are considered equal in the industrial example, although complex methods may profit from analytical microservices. Furthermore, the avoidance of losses thanks to the utilization of adequate solutions to support the production logistics functions, which would increase the benefit of accurate approaches, is not quantified in the industrial example.



# 8 Summary and Outlook

In chapter 8 the presented work is summarized. It does not only cover the proposed solution but also the context and factors that justify its creation. An outlook describing possible further activities to extend the applicability of the elaborated approach is also included.

## 8.1 Summary

In order to remain competitive, many manufacturers will have to migrate towards Personalized Production. Therefore, they have to manage an ever more complex production environment. As an answer to this problem, enterprises strive for flexibility. This, however, should not only make them technically capable of producing a wide range of products with varying features. It should also provide for an approach that is economically viable and satisfies the requirements of both, customers and manufacturing organization.

At the physical level, enterprises focus on the development and employment of technologies – machines, materials, and processes – that allow for a seamless adaptability. This is translated into product design, which pursues the twofold objective of appealing to customers while facilitating the manufacturing of products.

At the organizational level, upon which this work focuses, the components and processes of production logistics and order management are responsible for administrating and optimizing the manufacturing system. The underlying theory and techniques provide two important elements for enabling Personalized Production: a description of the functions involved (and their connection) and the key figures used for both, defining the objectives originating in the requirements of Personalized Production (e.g. delivery times comparable to the mass production) and measuring the performance of the system towards the

achievement of these objectives. To face the increasing complexity of organizational functions, the support of software tools is necessary. This is the task of production planning and control software and related applications.

The turbulence generated by Personalized Production, in addition to the existing difficulties in dynamic manufacturing systems, contributes to taking production logistics functions and their corresponding software implementations to their limit. Data analytics present a way to extend current functions beyond the existing deterministic approaches. This idea, however, gives rise to the following question: How can these analytical capabilities be applied and integrated into existing systems and business processes in an efficient and effective manner?

The proposed approach of analytical microservices presents an innovative solution. This does not only relate to the novelty of the IT architecture employed. Based on the core concept of *flexibility*, the idea combines the structural advantages of the microservices-based approach with the requirements and characteristics of data analytics and production logistics functions. All of this is done to achieve the creation of smart manufacturing systems, characterized by an integration of IT-based intelligence and business processes, a continuous adaptation, and the employment of domain knowledge. The proposed solution tries to overcome in this way the frequent dissociation found between the development of IT approaches and their application environment.

With software solutions, it should always be taken into account that the most complex tool is not necessarily the most adequate one for all cases. The validation shows that the utility of the approach rises with an increase in the complexity of the considered application scenario. However, as stated before, such a level of complexity is the trademark characteristic of Personalized Production. Furthermore, the real advantages of the solution are difficult to measure, as there are many benefits to a constantly adapting system which are mainly based on the avoidance of losses.



## 8.2 Outlook

This work presents an overview of the fundamentals of the proposed analytical microservices approach and its application to the challenge of Personalized Production. As such, several activities can be performed to deepen and extend the formulated idea.

First of all, the current work focuses mainly on the effectiveness and efficiency of the proposed solution under the conditions presented by Personalized Production. However, as briefly addressed, manufacturing systems today suffer from turbulences and disturbances. It would therefore be of interest to examine the suitability of analytical microservices under circumstances more ordinary than Personalized Production.

The scope of this work does not cover specific requirements of relevant operations research approaches – for example, the Simplex Method. Since such techniques suit current production optimization activities, it would be interesting to test the applicability of the designed solution and to eventually extend it as a result.

Another area of research could be the application of the solution to the process of data preparation. This offers a considerable potential, not only because of the effort employed in such activities, but also because of the possibilities to flexibly construct data cleansing processes and to reuse functionalities – thus reducing the costs incurred.

Within a research project called *AppAlytics* – financed by the German *Bundesministerium für Bildung und Forschung* (BMBF) – a prototype very similar to the proposed idea was developed. Not only the technical characteristics and challenges of the solution were analyzed, but also feedback from the different possible stakeholders – users and providers of production management software and data analytics – was gathered. More research should be done in order to develop a market-ready solution. It is advisable to perform such activities in collaboration with industrial partners from the possible stakeholder groups.

In the area of research, several activities towards the employment of artificial intelligence in production can benefit from the proposed approach. This would enable the creation of

cognitive manufacturing systems (Colangelo et al. 2019) where the application of this kind of intelligence should become ubiquitous.

## 9 Bibliography

**Aamodt et al. 1994**

Aamodt, Agnar; Plaza, Enric, 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* **7** (1), pp. 39-59  
DOI: 10.3233/AIC-1994-7104

**Adenowo et al. 2020**

Adenowo, Adetokunbo; Adenowo, Basirat, 2020. Software Engineering Methodologies: A Review of the Waterfall Model and Object- Oriented Approach. *International Journal of Scientific and Engineering Research* **4**, pp. 427-434

**Aickelin et al. 2011**

Aickelin, Uwe; Clark, Alistair, 2011. Heuristic optimisation. *Journal of the Operational Research Society* **62** (2), pp. 251-252  
DOI: 10.1057/jors.2010.160

**Al-Jarrah et al. 2015**

Al-Jarrah, Omar Y.; Yoo, Paul D.; Muhaidat, Sami; Karagiannidis, George K; Taha, Kamal, 2015. Efficient Machine Learning for Big Data: A Review. *Big Data Research* **2** (3), pp. 87-93  
DOI: 10.1016/j.bdr.2015.04.001

**Alpaydin 2016**

Alpaydin, Ethem, 2016. *Machine Learning: The New AI*. Cambridge: The MIT Press. The MIT Press Essential Knowledge series. ISBN 9780262529518

**Anand et al. 1998**

Anand, Sarabjot S.; Büchner, Alex G., 1998. *Decision support using data mining*. 1st edition. London u.a.: Financial Times Management. Financial Times management briefings : Information technology. ISBN 0-273-63269-8

**ANSI/ISA 95**

ANSI/ISA 95. *Enterprise-Control System Integration*.

- Arora et al. 2018** Arora, Nipun; Bell, Jonathan; Ivančić, Franjo; Kaiser, Gail; Ray, Baishakhi, 2018.  
Replay without recording of production bugs for service oriented applications.  
In: Huchard, Marianne; Kästner, Christian; Fraser, Gordon (Eds.): *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE 2018*, Montpellier, France, 03.09.2018-07.09.2018, pp. 452-463  
ISBN 9781450359375  
DOI: 10.1145/3238147.3238186
- Ashby 2015** Ashby, William Ross, 2015.  
*An introduction to cybernetics*.  
Mansfield Centre, CT: Martino Publishing.  
ISBN 978-1614277651
- Aytar et al. 2011** Aytar, Yusuf; Zisserman, Andrew, 2011.  
Tabula rasa: Model transfer for object category detection.  
In: *2011 International Conference on Computer Vision*, Barcelona, Spain, 06.11.2011-13.11.2011, pp. 2252-2259  
ISBN 978-1-4577-1102-2  
DOI: 10.1109/ICCV.2011.6126504
- Azevedo et al. 2008** Azevedo, Ana; Santos, Manuel Filipe, 2008.  
KDD, semma and CRISP-DM: A parallel overview.  
In: Weghorn, Hans; Abraham, Ajith P. (Eds.): *Proceedings of the IADIS European Conference on Data Mining*, Amsterdam, The Netherlands, 22.07.2008-27.07.2008, pp. 182-185  
ISBN 978-972-8924-63-8
- Bang et al. 2019** Bang, Seung Hwan; Ak, Ronay; Narayanan, Anantha; Lee, Y. Tina; Cho, Hyunbo, 2019. A survey on knowledge transfer for manufacturing data analytics.  
*Computers in Industry* **104**, pp. 116-130  
DOI: 10.1016/j.compind.2018.07.001
- Bartschat et al. 2019** Bartschat, Andreas; Reischl, Markus; Mikut, Ralf, 2019.  
Data mining tools.  
*Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **9** (4)  
DOI: 10.1002/widm.1309

- Batini et al. 2009** Batini, Carlo; Cappiello, Cinzia; Francalanci, Chiara; Maurino, Andrea, 2009. Methodologies for data quality assessment and improvement. *ACM Computing Surveys* **41** (3), pp. 1-52  
DOI: 10.1145/1541880.1541883
- Bauer et al. 2013** Bauer, Andreas; Günzel, Holger, 2013. *Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung*. Heidelberg: dpunkt.verlag.  
ISBN 978-3898647854
- Bauernhansl et al. 2014a** Bauernhansl, Thomas; Hompel, Michael ten; Vogel-Heuser, Birgit, 2014. *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Wiesbaden: Springer Fachmedien.  
ISBN 978-3-658-04681-1  
DOI: 10.1007/978-3-658-04682-8
- Bauernhansl et al. 2014b** Bauernhansl, Thomas; Schatz, Anja; Jäger, Jens, 2014. Komplexität bewirtschaften – Industrie 4.0 und die Folgen. *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* **109** (5), pp. 347-350  
DOI: 10.3139/104.111140
- Becker 2004** Becker, Jörg, 2004. *Handelsinformationssysteme: Domänenorientierte Einführung in die Wirtschaftsinformatik*. 1st edition. München: mi-Wirtschaftsbuch.  
ISBN 978-3-636-03144-0
- Becker et al. 2009** Becker, Jörg; Krcmar, Helmut; Niehaves, Björn, 2009. *Wissenschaftstheorie und gestaltungsorientierte Wirtschaftsinformatik*. Heidelberg: Physica-Verlag Heidelberg.  
ISBN 978-3-7908-2336-3  
DOI: 10.1007/978-3-7908-2336-3
- Belle et al. 2015** Belle, Ashwin; Thiagarajan, Raghuram; Soroushmehr, S. M. Reza; Navidi, Fatemeh; Beard, Daniel A; Najarian, Kayvan, 2015. Big Data Analytics in Healthcare. *BioMed research international* **2015**, p. 370194  
DOI: 10.1155/2015/370194

- Benz et al. 2011** Benz, Jochen; Höflinger, Markus, 2011.  
*Logistikprozesse mit SAP®: Eine anwendungsbezogene Einführung - Mit durchgehendem Fallbeispiel - Geeignet für SAP Version 4.6A bis ECC 6.0.*  
3rd edition.  
Wiesbaden: Vieweg+Teubner Verlag.  
ISBN 978-3-8348-1484-5  
DOI: 10.1007/978-3-8348-8119-9
- Bertsimas et al. 2020** Bertsimas, Dimitris; Kallus, Nathan, 2020. From Predictive to Prescriptive Analytics.  
*Management Science* **66** (3), pp. 1025-1044  
DOI: 10.1287/mnsc.2018.3253
- Bichindaritz 2015** Bichindaritz, Isabelle, 2015.  
Data Mining Methods for Case-Based Reasoning in Health Sciences.  
In: Kendall-Morwick, Joseph (Ed.): *The Twenty-Third International Conference on Case-Based Reasoning (ICCBR 2015) - Workshop Proceedings*, Frankfurt, Germany, 28.09.2015-30.09.2015, pp. 184-198
- Bishop 2006** Bishop, Christopher M., 2006.  
*Pattern recognition and machine learning.*  
New York, NY: Springer.  
Computer science.  
ISBN 978-0387-31073-2
- Bloomberg 2015** Bloomberg, Jason, 2015.  
*DevOps Insights into Conway's Law.*  
From: <https://inteltyx.com/2015/06/22/devops-insights-into-conways-law/>  
Viewed: 13.06.2019
- Bonham 2017** Bonham, Andrew, 2017.  
*Microservices—When to React Vs. Orchestrate.*  
From: <https://medium.com/capital-one-tech/microservices-when-to-react-vs-orchestrate-c6b18308a14c>  
Viewed: 14.06.2019

- Booch 2007** Booch, Grady, 2007. *Object-oriented analysis and design with applications*. 3rd edition. Upper Saddle River, NJ: Addison-Wesley. The Addison-Wesley object technology series. ISBN 0-201-89551-X
- Borse et al. 2019** Borse, Akash Ajay; Verma, Siddhant; Babu, Sasidhar; Kumar, Gardas Naresh, 2019. Service Oriented Architecture paradigm for Business Intelligence: A survey. *International Journal of Advance Research, Ideas and Innovations in Technology* **5** (1), pp. 196-199
- Bose 2009** Bose, Ranjit, 2009. Advanced analytics: opportunities and challenges. *Industrial Management & Data Systems* **109** (2), pp. 155-172  
DOI: 10.1108/02635570910930073
- Brachman et al. 1996** Brachman, Ronald J.; Anand, Tej, 1996. The Process of Knowledge Discovery in Databases: A Human-Centered Approach. In: Fayyad, Usama M. (Ed.): *Advances in knowledge discovery and data mining*. Menlo Park, Calif.: AAAI Press, pp. 37-58  
ISBN 9780262560979
- Breiman 2001** Breiman, Leo, 2001. Statistical Modeling The Two Cultures. *Statistical Science* **16** (3), pp. 199-231  
DOI: 10.1023/A:1010933404324
- Brodsky et al. 2015** Brodsky, Alexander; Shao, Guodong; Krishnamoorthy, Mohan; Narayanan, Anantha; Menasce, Daniel; Ak, Ronay, 2015. Analysis and optimization in smart manufacturing based on a reusable knowledge base for process performance models. In: Ho, Howard (Ed.): *2015 IEEE International Conference on Big Data*, Santa Clara, CA, USA, 29.10.2015-01.11.2015, pp. 1418-1427  
ISBN 978-1-4799-9926-2  
DOI: 10.1109/BigData.2015.7363902

- Brown 2014** Brown, Simon, 2014.  
*Distributed big balls of mud: If you can't build a monolith, what makes you think microservices are the answer?*  
From:  
[http://www.codingthearchitecture.com/2014/07/06/distributed\\_big\\_balls\\_of\\_mud.html](http://www.codingthearchitecture.com/2014/07/06/distributed_big_balls_of_mud.html)  
Viewed: 05.06.2019
- Bruce et al. 2017** Bruce, Peter; Bruce, Andrew, 2017.  
*Practical statistics for data scientists: 50 essential concepts*.  
Sebastopol, CA: O'Reilly Media.  
ISBN 978-1-491-95296-2
- Büchner et al. 1999** Büchner, Alex G; Hughes, John G; Bell, David A., 1999.  
Contextual Data and Domain Knowledge for Incorporation in Knowledge Discovery Systems.  
In: Goos, G; Hartmanis, J; van Leeuwen, J; Bouquet, Paolo; Benerecetti, Massimo; Serafini, Luciano; Brézillon, Patrick; Castellani, Francesca (Eds.): *Modeling and Using Context*.  
Berlin, Heidelberg: Springer, pp. 447-450  
ISBN 978-3-540-66432-1  
DOI: 10.1007/3-540-48315-2\_35
- Chahal et al. 2016** Chahal, Hemlata; Gulia, Preeti, 2016. Comprehensive Study of Open-Source Big Data Mining Tools.  
*International Journal of Artificial Intelligence and Knowledge Discovery* **6** (1), pp. 15-18
- Chan 2013** Chan, Joseph O., 2013. An Architecture for Big Data Analytics.  
*Communications of the IIMA* **13** (2), pp. 1-14  
DOI: 10.58729/1941-6687.1209
- Chertchom 2018** Chertchom, Prajak, 2018.  
A comparison study between data mining tools over regression methods: Recommendation for SMEs.  
In: *2018 5th International Conference on Business and Industrial Research (ICBIR)*, Bangkok, Thailand, 17.05.2018-18.05.2018, pp. 46-50  
ISBN 978-1-5386-5254-1  
DOI: 10.1109/ICBIR.2018.8391164



- Chien et al. 2005** Chien, Chen-Fu; Hsiao, Chih-Wei; Meng, Cheng; Hong, Kuo-Tong; Wang, Szu-Tsung, 2005. Cycle time prediction and control based on production line status and manufacturing data mining. In: *ISSM 2005*, San Jose, CA, USA, 13.09.2005-15.09.2005, pp. 327-330  
ISBN 0-7803-9143-8  
DOI: 10.1109/ISSM.2005.1513369
- Christensen et al. 2010** Christensen, Henrik Iskov; Kruijff, Geert-Jan M; Wyatt, Jeremy L., 2010. *Cognitive Systems*. Berlin, Heidelberg: Springer. Cognitive Systems Monographs 8. ISBN 978-3-642-11694-0  
DOI: 10.1007/978-3-642-11694-0
- Cleff 2015** Cleff, Thomas, 2015. *Deskriptive Statistik und Explorative Datenanalyse: Eine computergestützte Einführung mit Excel, SPSS und STATA*. Wiesbaden: Gabler Verlag. ISBN 978-3834947475
- Clyde et al. 2017** Clyde, Stephen; Lascano, Jorge Edison, 2017. Unifying Definitions for Modularity, Abstraction, and Encapsulation as a Step Toward Foundational Multi-Paradigm Software Engineering Principles. In: Lavazza, Luigi; Oberhauser, Roy; Koci, Radek; Clyde, Stephen (Eds.): *ICSEA 2017*, Athens, Greece, 08.10.2017-12.10.2017, pp. 105-113  
ISBN 978-1-61208-590-6
- Colangelo et al. 2016a** Colangelo, Eduardo; Pira, Axel; Noack, Anika; Bauernhansl, Thomas, 2016. Schritte zu Big Data in der Produktion. *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* **111** (12), pp. 851-854  
DOI: 10.3139/104.111635
- Colangelo et al. 2016b** Colangelo, Eduardo; Bauernhansl, Thomas, 2016. Usage of Analytical Services in Industry Today and Tomorrow. *Procedia CIRP* **57**, pp. 276-280  
DOI: 10.1016/j.procir.2016.11.048

- Colangelo et al. 2018** Colangelo, Eduardo; Kröger, Torsten; Bauernhansl, Thomas, 2018. Substitution and Complementation of Production Management Functions with Data Analytics.  
*Procedia CIRP* **72**, pp. 191-196  
DOI: 10.1016/j.procir.2018.03.145
- Colangelo et al. 2019** Colangelo, Eduardo; Hartleif, Silke; Kroger, Torsten; Bauernhansl, Thomas, 2019.  
A Service-Oriented Approach for the Cognitive Factory — A Position Paper.  
In: *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, Okinawa, Japan, 11.02.2019-13.02.2019, pp. 540-542  
ISBN 978-1-5386-7822-0  
DOI: 10.1109/ICAIIIC.2019.8668990
- Colfer et al. 2016** Colfer, Lyra J.; Baldwin, Carliss Y., 2016. The mirroring hypothesis: theory, evidence, and exceptions.  
*Industrial and Corporate Change* **25** (5), pp. 709-738  
DOI: 10.1093/icc/dtw027
- Coniglio et al. 2018** Coniglio, Stefano; Koster, Arie M. C. A.; Spiekermann, Nils, 2018. Lot sizing with storage losses under demand uncertainty.  
*Journal of Combinatorial Optimization* **36** (3), pp. 763-788  
DOI: 10.1007/s10878-017-0147-8
- Conway 1968** Conway, Melvin E., 1968. How Do Committees Invent?  
*Datamation* **14** (4), pp. 28-31
- Cos Juez et al. 2010** Cos Juez, Francisco Javier de; García Nieto, Paulino Jose; Martínez Torres, Javier; Taboada Castro, Javier, 2010. Analysis of lead times of metallic components in the aerospace industry through a supported vector machine model.  
*Mathematical and Computer Modelling* **52** (7-8), pp. 1177-1184  
DOI: 10.1016/j.mcm.2010.03.017

- D'Amore 2015** D'Amore, Jean, 2015.  
*Scaling Microservices with an Event Stream*.  
From:  
<https://www.thoughtworks.com/de/insights/blog/scaling-microservices-event-stream>  
Viewed: 14.06.2019
- Daya 2015** Daya, Shahir, 2015.  
*Microservices from theory to practice: Creating applications in IBM Bluemix using the microservices approach*.  
1st edition.  
IBM redbooks.  
ISBN 9780738440811
- Dedić et al. 2016** Dedić, Nedim; Stanier, Clare, 2016.  
Measuring the Success of Changes to Existing Business Intelligence Solutions to Improve Business Intelligence Reporting.  
In: Tjoa, A. M; Xu, Li D; Raffai, Maria; Novak, Niina M. (Eds.): *Research and Practical Issues of Enterprise Information Systems*.  
Cham: Springer International Publishing, pp. 225-236  
ISBN 978-3-319-49943-7  
DOI: 10.1007/978-3-319-49944-4\_17
- Dedić et al. 2017** Dedić, Nedim; Stanier, Clare, 2017.  
Towards Differentiating Business Intelligence, Big Data, Data Analytics and Knowledge Discovery.  
In: Piazzolo, Felix; Geist, Verena; Brehm, Lars; Schmidt, Rainer (Eds.): *Innovations in Enterprise Information Systems Management and Engineering*.  
Cham, s.l.: Springer International Publishing, pp. 114-122  
ISBN 978-3-319-58800-1  
DOI: 10.1007/978-3-319-58801-8\_10
- Dittmar 2004** Dittmar, Carsten, 2004.  
*Knowledge Warehouse: Ein integrativer Ansatz des Organisationsgedächtnisses und die computergestützte Umsetzung auf Basis des Data Warehouse-Konzepts*.  
Wiesbaden: Deutscher Universitätsverlag.  
ISBN 978-3824481262  
DOI: 10.1007/978-3-322-81770-9

- Dowalil 2018** Dowalil, Herbert, 2018.  
*Grundlagen des modularen Softwareentwurfs: Der Bau langlebiger Mikro- und Makro-Architekturen wie Microservices und SOA 2.0.*  
München: Hanser.  
ISBN 978-3446455092  
DOI: 10.3139/9783446456006
- Dragoni et al. 2017** Dragoni, Nicola; Giallorenzo, Saverio; Lafuente, Alberto Lluçh; Mazzara, Manuel; Montesi, Fabrizio; Mustafin, Ruslan; Safina, Larisa, 2017.  
Microservices: Yesterday, Today, and Tomorrow.  
In: Mazzara, Manuel; Meyer, Bertrand (Eds.): *Present and Ulterior Software Engineering.*  
Cham: Springer International Publishing, pp. 195-216  
ISBN 978-3-319-67424-7  
DOI: 10.1007/978-3-319-67425-4\_12
- Duan et al. 2003** Duan, Kaibo; Keerthi, S.Sathiy; Poo, Aun Neow, 2003.  
Evaluation of simple performance measures for tuning SVM hyperparameters.  
*Neurocomputing* **51**, pp. 41-59  
DOI: 10.1016/S0925-2312(02)00601-X
- Dürr 2013** Dürr, Peter, 2013.  
*Modell zur Bewertung der Effizienz der IT-Unterstützung im Auftragsabwicklungsprozess von produzierenden KMU.*  
Stuttgart: Fraunhofer-Verl.  
Stuttgarter Beiträge zur Produktionsforschung 16.  
Stuttgart, Univ., Diss., 2013.  
ISBN 978-3-8396-0579-0
- Eden et al. 2006** Eden, Amnon H.; Mens, Tom, 2006. Measuring software flexibility.  
*IEE Proceedings - Software* **153** (3), p. 113  
DOI: 10.1049/ip-sen:20050045
- ElMaraghy et al. 2009** ElMaraghy, Hoda A.; Wiendahl, Hans-Peter, 2009.  
Changeability – An Introduction.  
In: ElMaraghy, Hoda A. (Ed.): *Changeable and Reconfigurable Manufacturing Systems.*  
London: Springer, pp. 3-24  
ISBN 978-1-84882-066-1  
DOI: 10.1007/978-1-84882-067-8\_1

- ElMaraghy 2009** ElMaraghy, Hoda A., 2009. Changing and Evolving Products and Systems – Models and Enablers. In: ElMaraghy, Hoda A. (Ed.): *Changeable and Reconfigurable Manufacturing Systems*. London: Springer, pp. 25-45  
ISBN 978-1-84882-066-1  
DOI: 10.1007/978-1-84882-067-8\_2
- Erl et al. 2017** Erl, Thomas; Merson, Paulo; Stoffers, Roger, 2017. *Service-oriented architecture: Analysis and design for services and microservices*. Boston: Prentice Hall Service Tech Press. Prentice Hall service technology series from Thomas Erl. ISBN 978-0133858587
- Ester et al. 2000** Ester, Martin; Sander, Jörg, 2000. *Knowledge Discovery in Databases: Techniken und Anwendungen*. Berlin, Heidelberg, s.l.: Springer Berlin Heidelberg. ISBN 9783540673286  
DOI: 10.1007/978-3-642-58331-5
- Evans et al. 2012** Evans, James R.; Lindner, Carl H., 2012. Business Analytics: The Next Frontier for Decision Sciences. *Decision Line* **43** (2), pp. 4-6
- Evans 2015** Evans, Eric, 2015. *Domain-driven design reference: Definitions and pattern summaries*. Indianapolis: Dog Ear Publishing. ISBN 9781457501197
- Evgeniou et al. 2002** Evgeniou, Theodoros; Pontil, Massimiliano, 2002. Support Vector Machines with Clustering for Training with Very Large Datasets. In: Goos, G; Hartmanis, J; van Leeuwen, J; Vlahavas, Ioannis P; Spyropoulos, Constantine D. (Eds.): *Methods and Applications of Artificial Intelligence*. Berlin, Heidelberg: Springer, pp. 346-354  
ISBN 978-3-540-43472-6  
DOI: 10.1007/3-540-46014-4\_31

- Fayyad 1996** Fayyad, Usama M. (Ed.), 1996. *Advances in knowledge discovery and data mining*. Menlo Park, Calif.: AAAI Press. ISBN 9780262560979
- Fayyad et al. 1996** Fayyad, Usama; Piatetsky-Shapiro, Gregory; Smyth, Padhraic, 1996. From Data Mining to Knowledge Discovery in Databases. *AI Magazine* **17** (3), pp. 37-54  
DOI: 10.1609/aimag.v17i3.1230
- Fernández et al. 2014** Fernández, Alberto; del Río, Sara; López, Victoria; Bawakid, Abdullah; del Jesus, María J.; Benítez, José M.; Herrera, Francisco, 2014. Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **4** (5), pp. 380-409  
DOI: 10.1002/widm.1134
- Fischer et al. 2019** Fischer, Jonas; Springer, Patrick; Fulga-Beising, Bogdana Simina; Abu El-Qomsan, Khalid, 2019. Enabler für die personalisierte Produktion, Ein Beispiel der automatisierten, additiven Fertigung von maßgeschneiderten Produkten. *wt Werkstattstechnik online* **109** (3), pp. 179-183  
DOI: 10.37544/1436-4980-2019-03-77
- Fowler 2017** Fowler, Susan J., 2017. *Production-ready microservices: Building standardized systems across an engineering organization*. Sebastopol, CA: O'Reilly Media. ISBN 978-1-491-96597-9
- Freitag et al. 2015** Freitag, Michael; Kück, Mirko; Ait Alla, Abderrahim; Lütjen, Michael, 2015. Potenziale von Data Science in Produktion und Logistik: Teil 1 - Eine Einführung in aktuelle Ansätze der Data Science. *Industrie 4.0 Management* **31** (5), pp. 22-26
- Furht et al. 2016** Furht, Borko; Villanustre, Flavio (Eds.), 2016. *Big Data Technologies and Applications*. Cham: Springer International Publishing. ISBN 978-3-319-44548-9  
DOI: 10.1007/978-3-319-44550-2

- Gabriel et al. 2011** Gabriel, Roland; Gluchowski, Peter; Pastwa, Alexander, 2011.  
*Data warehouse & data mining*.  
1st edition.  
Herdecke, Witten: W3L-Verl.  
Informatik.  
ISBN 978-3937137667
- Gandomi et al. 2015** Gandomi, Amir; Haider, Murtaza, 2015. Beyond the hype: Big data concepts, methods, and analytics.  
*International Journal of Information Management* **35** (2), pp. 137-144  
DOI: 10.1016/j.ijinfomgt.2014.10.007
- García-Borgoñón et al. 2014** García-Borgoñón, Laura; Barcelona, Miguel Ángel; García-García, Julián Alberto; Alba, Manuel; Escalona, María José, 2014. Software process modeling languages: A systematic literature review.  
*Information and Software Technology* **56** (2), pp. 103-116  
DOI: 10.1016/j.infsof.2013.10.001
- Gartner 2019** Gartner, 2019.  
*IT Glossary*.  
From: <https://www.gartner.com/it-glossary/>  
Viewed: 23.04.2019
- Gauthier et al. 1970** Gauthier, Richard L.; Ponto, Stephen D., 1970.  
*Designing systems programs*.  
Englewood Cliffs, N.J.: Prentice-Hall.  
Prentice-Hall series in automatic computation.  
ISBN 9780132019620
- George et al. 2014** George, Gerard; Haas, Martine R.; Pentland, Alex, 2014. Big Data and Management.  
*Academy of Management Journal* **57** (2), pp. 321-326  
DOI: 10.5465/amj.2014.4002
- Gepperth et al. 2016** Gepperth, Alexander; Hammer, Barbara, 2016. Incremental learning algorithms and applications.  
In: Verleysen, Michel (Ed.): *24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium, 27.04.2016-29.04.2016, pp. 357-368  
ISBN 978-287587027-8

- Gläßner et al. 1991** Gläßner, Jürgen; Nyhuis, Peter, 1991. Zielkonflikt lösen – Durchlauforientierte Losbildung im Vergleich zu konventionellen Verfahren. *Maschinenmarkt* **97** (8), pp. 60-64
- Goll 2018** Goll, Joachim, 2018. *Entwurfsprinzipien und Konstruktionskonzepte der Softwaretechnik: Strategien für schwach gekoppelte, korrekte und stabile Software*. Wiesbaden: Springer Vieweg. ISBN 978-3658200541 DOI: 10.1007/978-3-658-20055-8
- Goos et al. 2002** Goos, Gerhard; Hartmanis, Juris; van Leeuwen, Jan; Naumann, Felix, 2002. *Quality-Driven Query Answering for Integrated Information Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg 2261. ISBN 978-3-540-43349-1 DOI: 10.1007/3-540-45921-9
- Götze 2008** Götze, Uwe, 2008. *Investitionsrechnung*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-78872-0 DOI: 10.1007/978-3-540-78873-7
- Gröger 2018** Gröger, Christoph, 2018. Building an Industry 4.0 Analytics Platform. *Datenbank-Spektrum* **18** (1), pp. 5-14 DOI: 10.1007/s13222-018-0273-1
- Guedes et al. 2006** Guedes, Dorgival; Meira, Wagner; Ferreira, Renato, 2006. Anteaater: A Service-Oriented Architecture for High-Performance Data Mining. *IEEE Internet Computing* **10** (4), pp. 36-43 DOI: 10.1109/MIC.2006.69
- Günther et al. 2016** Günther, Hans-Otto; Tempelmeier, Horst, 2016. *Produktion und Logistik: Supply Chain und Operations Management*. 12th edition. Norderstedt: BoD - Books on Demand. ISBN 978-3741209628



- Günther et al. 2019** Günther, Lisa C; Colangelo, Eduardo; Wiendahl, Hans-Hermann; Bauer, Christian, 2019. Data quality assessment for improved decision-making: a methodology for small and medium-sized enterprises. *Procedia Manufacturing* **29**, pp. 583-591  
DOI: 10.1016/j.promfg.2019.02.114
- H.-H. Wiendahl 2002** Wiendahl, Hans-Hermann, 2002. *Situative Konfiguration des Auftragsmanagements im turbulenten Umfeld*. Heimsheim: Jost-Jetter. IPA-IAO-Forschung und -Praxis 358. Stuttgart, Univ., Diss., 2002. ISBN 3-931388-87-5
- H.-H. Wiendahl et al. 2005** Wiendahl, Hans-Hermann; Cieminski, Gregor von; Wiendahl, Hans-Peter, 2005. Stumbling blocks of PPC: Towards the holistic configuration of PPC systems. *Production Planning & Control* **16** (7), pp. 634-651  
DOI: 10.1080/09537280500249280
- H.-H. Wiendahl 2007** Wiendahl, Hans-Hermann, 2007. Turbulence Germs and their Impact on Planning and Control – Root Causes and Solutions for PPC Design. *CIRP Annals* **56** (1), pp. 443-446  
DOI: 10.1016/j.cirp.2007.05.106
- H.-H. Wiendahl 2009** Wiendahl, Hans-Hermann, 2009. Adaptive Production Planning and Control – Elements and Enablers of Changeability. In: ElMaraghy, Hoda A. (Ed.): *Changeable and Reconfigurable Manufacturing Systems*. London: Springer, pp. 197-212  
ISBN 978-1-84882-066-1  
DOI: 10.1007/978-1-84882-067-8\_11
- H.-H. Wiendahl 2012** Wiendahl, Hans-Hermann, 2012. *Auftragsmanagement der industriellen Produktion: Grundlagen, Konfiguration, Einführung*. Berlin, Heidelberg: Springer. ISBN 978-3-642-19148-0  
DOI: 10.1007/978-3-642-19149-7

- H.-H. Wiendahl 2020** Wiendahl, Hans-Hermann, 2020.  
Auftragsmanagement.  
In: Bauernhansl, Thomas (Ed.): *Fabrikbetriebslehre 1*.  
Berlin, Heidelberg: Springer, pp. 193-294  
ISBN 978-3-662-44537-2  
DOI: 10.1007/978-3-662-44538-9\_7
- H.-P. Wiendahl 1997** Wiendahl, Hans-Peter, 1997.  
*Fertigungsregelung: Logistische Beherrschung von  
Fertigungsabläufen auf Basis des Trichtermodells*.  
München: Hanser.  
ISBN 978-3-446-19084-9
- H.-P. Wiendahl et al. 2020** Wiendahl, Hans-Peter; Wiendahl, Hans-Hermann,  
2020.  
*Betriebsorganisation für Ingenieure: Mit 279  
Abbildungen*.  
9th edition.  
ISBN 978-3446446618
- Hackstein 1989** Hackstein, Rolf, 1989.  
*Produktionsplanung und -steuerung (PPS): Ein  
Handbuch für die Betriebspraxis*.  
2nd edition.  
Düsseldorf: VDI-Verl.  
ISBN 3184009246
- Hajjat et al. 2010** Hajjat, Mohammad; Sun, Xin; Sung, Yu-Wei Eric;  
Maltz, David; Rao, Sanjay; Sripanidkulchai, Kunwadee;  
Tawarmalani, Mohit, 2010.  
Cloudward bound.  
In: Kalyanaraman, Shiv (Ed.): *Proceedings of the ACM  
SIGCOMM 2010 conference*, New Delhi, India,  
30.08.2010-03.09.2010, pp. 243-254  
ISBN 9781450302012  
DOI: 10.1145/1851182.1851212
- Hamedtavasoli et al. 2021** Hamedtavasoli, Hosein; Delic, Nedim; Blanco, Quinn,  
2021. Transforming ERP from a system of registration  
to a system of analytics.  
*compact\_ 48* (3), pp. 32-37

- Han et al. 2012** Han, Jiawei; Kamber, Micheline; Pei, Jian, 2012. *Data mining: Concepts and techniques*. 3rd edition. Amsterdam: Elsevier/Morgan Kaufmann. The Morgan Kaufmann series in data management systems. ISBN 978-0-12-381479-1
- Hand 1999** Hand, David J., 1999. Statistics and data mining. *ACM SIGKDD Explorations Newsletter* **1** (1), pp. 16-19  
DOI: 10.1145/846170.846171
- Hand et al. 2001** Hand, David J.; Mannila, Heikki; Smyth, Padhraic, 2001. *Principles of data mining*. Cambridge Mass. u.a.: MIT Press. Adaptive computation and machine learning. ISBN 978-0-262-08290-7
- Hayashi et al. 2012** Hayashi, Takafumi; Kara, Atushi; Miyazaki, Toshiaki; Iwase, Jiro; Fukuhara, Hideyuki; Saburi, Tetsu; Hisada, Masayuki, 2012. Coping with the Complexity of SOA Systems with Message Forensics. In: *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, Fukuoka, Japan, 26.03.2012-29.03.2012, pp. 732-737  
ISBN 978-1-4673-0867-0  
DOI: 10.1109/WAINA.2012.99
- Heger et al. 2015** Heger, Jens; Hildebrandt, Torsten; Scholz-Reiter, Bernd, 2015. Dispatching rule selection with Gaussian processes. *Central European Journal of Operations Research* **23** (1), pp. 235-249  
DOI: 10.1007/s10100-013-0322-7
- Hevner et al. 2004** Hevner, Alan; March, Salvatore; Park, Jinsoo; Ram, Sudha, 2004. Design Science in Information Systems Research. *MIS Quarterly* **28** (1), pp. 75-105

- Hoekstra et al. 1992** Hoekstra, Sjoerd; Romme, Jac (Eds.), 1992. *Integral logistic structures: Developing customer-oriented goods flow*. London, New York: McGraw-Hill. ISBN 9780077075521
- Hu et al. 2011** Hu, Shixin Jack; Ko, Jeonghan; Weyand, Lars; ElMaraghy, Hoda A; Lien, Terje; Koren, Yoram; Bley, Helmut; Chryssolouris, George; Nasr, Nabil; Shpitalni, Moshe, 2011. Assembly system design and operations for product variety. *CIRP Annals* **60** (2), pp. 715-733  
DOI: 10.1016/j.cirp.2011.05.004
- Hu 2013** Hu, Shixin Jack, 2013. Evolving Paradigms of Manufacturing: From Mass Production to Mass Customization and Personalization. *Procedia CIRP* **7**, pp. 3-8  
DOI: 10.1016/j.procir.2013.05.002
- Huang et al. 2017** Huang, Yutao; Ma, Xiaoqiang; Fan, Xiaoyi; Liu, Jiangchuan; Gong, Wei, 2017. When deep learning meets edge computing. In: *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, Toronto, Canada, 10.10.2017-13.10.2017, pp. 1-2  
ISBN 978-1-5090-6501-1  
DOI: 10.1109/ICNP.2017.8117585
- IEEE 1471-2000** IEEE 1471-2000. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*.
- IEEE 610.12-1990** IEEE 610.12-1990. *IEEE Standard Glossary of Software Engineering Terminology*.
- Inmon 2005** Inmon, William H., 2005. *Building the data warehouse*. 4th edition. Indianapolis: Wiley Pub. ISBN 978-0-7645-9944-6

- ISO/IEC 13249-6:2006** ISO/IEC 13249-6:2006.  
*Information technology - Database languages - SQL multimedia and application packages: Part 6: Data Mining*
- ISO/IEC 18384-2:2016** ISO/IEC 18384-2:2016.  
*Information technology — Reference Architecture for Service Oriented Architecture (SOA RA): Part 2: Reference Architecture for SOA Solutions*
- Janeczko et al. 2013** Janeczko, Jordan; Caminel, Thierry; Roberts, Matthew, 2013.  
*Data Analytics as a Service: unleashing the power of Cloud and Big Data.*  
From:  
<https://atos.net/content/dam/global/documents/your-business/atos-ascent-white-paper-daaas.pdf>  
Viewed: 05.04.2021
- Jarwar et al. 2018** Jarwar, Muhammad Aslam; Ali, Sajjad; Chong, Ilyoung, 2018.  
Exploring Web Objects enabled Data-Driven Microservices for E-Health Service Provision in IoT Environment.  
In: *ICTC 2018 : the 9th International Conference on ICT Convergence*, Jeju, South Korea, 17.10.2018-19.10.2018, pp. 112-117  
ISBN 978-1-5386-5041-7  
DOI: 10.1109/ICTC.2018.8539684
- Ji et al. 2017** Ji, Wei; Wang, Lihui, 2017. Big data analytics based fault prediction for shop floor scheduling.  
*Journal of Manufacturing Systems* **43**, pp. 187-194  
DOI: 10.1016/j.jmsy.2017.03.008
- Jones 2017** Jones, Tim, 2017.  
*Models for machine learning: Explore the ideas behind machine learning models and some key algorithms used for each.*  
From: <https://developer.ibm.com/articles/cc-models-machine-learning/>  
Viewed: 20.08.2020

- Kacfeh Emani et al. 2015** Kacfeh Emani, Cheikh; Cullot, Nadine; Nicolle, Christophe, 2015. Understandable Big Data: A survey. *Computer Science Review* **17**, pp. 70-81  
DOI: 10.1016/j.cosrev.2015.05.002
- Kasanoff 2009** Kasanoff, Bruce, 2009. *Personal=Smarter*". Interview in the Blog "*Mass Customization & Open Innovation*".  
From: [https://mass-customization.blogs.com/mass\\_customization\\_open\\_i/2009/04/interview-bruce-kasanoff-of-nowpossiblecom-on-personalsmarter.html](https://mass-customization.blogs.com/mass_customization_open_i/2009/04/interview-bruce-kasanoff-of-nowpossiblecom-on-personalsmarter.html)  
Viewed: 01.05.2022
- Kaur et al. 2020** Kaur, Avinash; Singh, Parminder; Nayyar, Anand, 2020. Fog Computing: Building a Road to IoT with Fog Analytics.  
In: Tanwar, Sudeep (Ed.): *Fog Data Analytics for IoT Applications*.  
Singapore: Springer Singapore, pp. 59-78  
ISBN 978-981-15-6043-9  
DOI: 10.1007/978-981-15-6044-6\_4
- Kehal 2020** Kehal, Mounir, 2020. *Data Analytics in Marketing, Entrepreneurship, and Innovation*.  
Milton: Auerbach Publishers Incorporated.  
ISBN 978-0367184834
- Keller et al. 2018** Keller, Markus; Baum, Gabriela; Schweizer, Marion; Bürger, Frank; Gommel, Udo; Bauernhansl, Thomas, 2018-2018. Optimized Robot Systems for Future Aseptic Personalized Mass Production.  
In: Wang, L. e.a. (Ed.): *51st CIRP Conference on Manufacturing Systems (CIRP CMS 2018)*, Stockholm, Sweden, 16.05.2018-18.05.2018, pp. 303-309  
DOI: 10.1016/j.procir.2018.03.066

- 
- Kiczales et al. 1997** Kiczales, Gregor; Lamping, John; Mendhekar, Anurag; Maeda, Chris; Lopes, Cristina; Loingtier, Jean-Marc; Irwin, John, 1997.  
Aspect-oriented programming.  
In: Akşit, Mehmet; Matsuoka, Satoshi (Eds.): *ECOOP'97 - Object-Oriented Programming*, Jyväskylä, Finland, 09.06.1997-13.06.1997, pp. 220-242  
ISBN 978-3-540-63089-0  
DOI: 10.1007/BFb0053381
- Kimball 2009** Kimball, Ralph, 2009.  
*The data warehouse lifecycle toolkit: Practical techniques for building data warehouse and business intelligence systems.*  
2nd edition.  
Indianapolis: Wiley.  
ISBN 978-0470149775
- Kimball et al. 2013** Kimball, Ralph; Ross, Margy, 2013.  
*The data warehouse toolkit: The definitive guide to dimensional modeling.*  
3rd edition.  
Indianapolis: Wiley.  
ISBN 978-1-118-53080-1
- Kletti 2007** Kletti, Jürgen, 2007.  
*Konzeption und Einführung von MES-Systemen.*  
1st edition.  
Berlin, Heidelberg: Springer.  
ISBN 3540343091
- Koren 2010** Koren, Yoram, 2010.  
*The global manufacturing revolution: Product-process-business integration and reconfigurable systems.*  
Hoboken, NJ: Wiley a John Wiley & Sons Inc.  
Wiley series in systems engineering and management.  
ISBN 978-0470583777  
DOI: 10.1002/9780470618813
- Koren et al. 2015** Koren, Yoram; Shpitalni, Moshe; Gu, Peihua; Hu, Shixin, 2015. Product Design for Mass-Individualization.  
*Procedia CIRP* **36**, pp. 64-71  
DOI: 10.1016/j.procir.2015.03.050

- Koste et al. 1999** Koste, Lori L.; Malhotra, Manoj K., 1999. A theoretical framework for analyzing the dimensions of manufacturing flexibility. *Journal of Operations Management* **18** (1), pp. 75-93  
DOI: 10.1016/S0272-6963(99)00010-8
- Koste et al. 2004** Koste, Lori L.; Malhotra, Manoj K.; Sharma, Subhash, 2004. Measuring dimensions of manufacturing flexibility. *Journal of Operations Management* **22** (2), pp. 171-196  
DOI: 10.1016/j.jom.2004.01.001
- Kozjek et al. 2018** Kozjek, Dominik; Vrabič, Rok; Rihtaršič, Borut; Butala, Peter, 2018. Big data analytics for operations management in engineer-to-order manufacturing. *Procedia CIRP* **72**, pp. 209-214  
DOI: 10.1016/j.procir.2018.03.098
- Kraker 2013** Kraker, Peter, 2013. *The models vs. patterns problem*.  
From:  
<https://science20.wordpress.com/2013/03/04/the-models-vs-patterns-problem/>  
Viewed: 26.04.2019
- Kraker et al. 2013** Kraker, Peter; Dennerlein, Sebastian, 2013. Towards a Model of Interdisciplinary Teamwork for Web Science: What can Social Theory Contribute? In: *Web Science 2013 Workshop: Harnessing the Power of Social Theory for Web Science*, Paris, France, 01.05.2013
- Krieger 2018** Krieger, Winfried, 2018. *Produktionslogistik*.  
From:  
<https://wirtschaftslexikon.gabler.de/definition/produktionslogistik-43280/version-266611>  
Viewed: 06.01.2019



- Kristianto et al. 2013** Kristianto, Yohanes; Helo, Petri; Jiao, Roger J., 2013. Mass customization design of engineer-to-order products using Benders' decomposition and bi-level stochastic programming. *Journal of Intelligent Manufacturing* **24** (5), pp. 961-975  
DOI: 10.1007/s10845-012-0692-z
- Kühnapfel 2019** Kühnapfel, Jörg B., 2019. *Nutzwertanalysen in Marketing und Vertrieb*. Wiesbaden: Springer Fachmedien. ISBN 978-3-658-25163-5  
DOI: 10.1007/978-3-658-25164-2
- Kumar 2007** Kumar, Ashok, 2007. From mass customization to mass personalization: A strategic transformation. *International Journal of Flexible Manufacturing Systems* **19** (4), pp. 533-547  
DOI: 10.1007/s10696-008-9048-6
- Kusiak 2017** Kusiak, Andrew, 2017. Smart manufacturing must embrace big data. *Nature* **544** (7648), pp. 23-25  
DOI: 10.1038/544023a
- Leroy et al. 2011** Leroy, Jonny; Simons, Matt, 2011. *Dealing with creaky legacy platforms*. From: <http://jonnyleroy.com/2011/02/03/dealing-with-creaky-legacy-platforms/>  
Viewed: 01.05.2022
- Lever et al. 2016** Lever, Jake; Krzywinski, Martin; Altman, Naomi, 2016. Model selection and overfitting. *Nature Methods* **13** (9), pp. 703-704  
DOI: 10.1038/nmeth.3968
- Liberatore et al. 2011** Liberatore, Matthew; Luo, Wenhong, 2011. INFORMS and the Analytics Movement: The View of the Membership. *Interfaces* **41** (6), pp. 578-589  
DOI: 10.1287/inte.1110.0599
- Lin 2017** Lin, Jimmy, 2017. The Lambda and the Kappa. *IEEE Internet Computing* **21** (5), pp. 60-66  
DOI: 10.1109/MIC.2017.3481351

- Lingitz et al. 2018** Lingitz, Lukas; Gallina, Viola; Ansari, Fazel; Gyulai, Dávid; Pfeiffer, András; Sihn, Wilfried; Monostori, László, 2018. Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer.  
*Procedia CIRP* **72**, pp. 1051-1056  
DOI: 10.1016/j.procir.2018.03.148
- Lödding 2008** Lödding, Hermann, 2008.  
*Verfahren der Fertigungssteuerung*.  
Berlin, Heidelberg: Springer.  
ISBN 978-3-540-76859-3  
DOI: 10.1007/978-3-540-76860-9
- Lu et al. 2020** Lu, Wanjie; Xu, Qing; Lan, Chaozhen; Lyu, Liang; Zhou, Yang; Shi, Qunshan; Zhao, Yinghao, 2020.  
Microservice-Based Platform for Space Situational Awareness Data Analytics.  
*International Journal of Aerospace Engineering* **2020**, pp. 1-22  
DOI: 10.1155/2020/8149034
- Malkawi et al. 2020** Malkawi, Rami; Saifan, Ahmad A.; Alhendawi, Nouh; Banilsmael, Alaa, 2020. Data Mining Tools Evaluation Based on their Quality Attributes.  
*International Journal of Advanced Science and Technology* **29** (3), pp. 13867-13890
- Mandel 2012** Mandel, Jörg, 2012.  
*Modell zur Gestaltung von Build-to-Order-Produktionsnetzwerken*.  
Stuttgart: Fraunhofer Verl.  
Stuttgarter Beiträge zur Produktionsforschung 2.  
Stuttgart, Univ., Diss., 2011.  
ISBN 978-3-8396-0434-2
- Marbán et al. 2008** Marbán, Oscar; Menasalvas, Ernestina; Fernández-Baizán, Covadonga, 2008. A cost model to estimate the effort of data mining projects (DMCoMo).  
*Information Systems* **33** (1), pp. 133-150  
DOI: 10.1016/j.is.2007.07.004

- Martin 2018** Martin, Robert C., 2018.  
*Clean architecture: A craftsman's guide to software structure and design*.  
Boston: Prentice Hall.  
Robert C. Martin series.  
ISBN 978-0134494166
- Marz et al. 2015** Marz, Nathan; Warren, James, 2015.  
*Big data: Principles and best practices of scalable real-time data systems*.  
Shelter Island, NY: Manning.  
ISBN 9781617290343
- Mather et al. 1977** Mather, Hal; Plossl, George, 1977.  
Priority fixation versus throughput planning.  
In: *Proceedings of the APICS International Conference*,  
Cleveland, OH, USA, pp. 27-51
- Mazzara et al. 2020** Mazzara, Manuel; Bucchiarone, Antonio; Dragoni, Nicola; Rivera, Victor, 2020.  
Size Matters: Microservices Research and Applications.  
In: Bucchiarone, Antonio; Dragoni, Nicola; Dustdar, Schahram; Lago, Patricia; Mazzara, Manuel; Rivera, Victor; Sadovykh, Andrey (Eds.): *Microservices*.  
Cham: Springer International Publishing, pp. 29-42  
ISBN 978-3-030-31645-7  
DOI: 10.1007/978-3-030-31646-4\_2
- Mell et al. 2011** Mell, Peter; Grance, Timothy, 2011.  
*The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology*.  
From: <https://csrc.nist.gov/publications/detail/sp/800-145/final>  
Viewed: 05.04.2021
- Mentzer et al. 2008** Mentzer, John T.; Stank, Theodore P.; Esper, Terry L., 2008. Supply Chain Management and its Relationship to Logistics, Marketing, Production, and Operations Management.  
*Journal of Business Logistics* **29** (1), pp. 31-46  
DOI: 10.1002/j.2158-1592.2008.tb00067.x

- Mikut et al. 2011** Mikut, Ralf; Reischl, Markus, 2011. Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **1** (5), pp. 431-443  
DOI: 10.1002/widm.24
- Miles et al. 2006** Miles, Russell; Hamilton, Kim, 2006. *Learning UML 2.0*. 1st edition. Beijing, Sebastopol, California: O'Reilly. ISBN 978-0-59-600982-3
- Mishra et al. 2021** Mishra, Swati; Rzeszotarski, Jeffrey M., 2021. Designing Interactive Transfer Learning Tools for ML Non-Experts. In: Kitamura, Yoshifumi; Quigley, Aaron; Isbister, Katherine; Igarashi, Takeo; Bjørn, Pernille; Drucker, Steven (Eds.): *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, Yokohama, Japan, 08.05.2021-13.05.2021, pp. 1-15  
ISBN 9781450380966  
DOI: 10.1145/3411764.3445096
- Mitchell 1997** Mitchell, Tom M., 1997. *Machine learning*. New York, NY: McGraw-Hill. McGraw-Hill series in computer science. ISBN 0070428077
- Motwani et al. 2007** Motwani, Rajeev; Raghavan, Prabhakar, 2007. *Randomized algorithms*. 9th edition. Cambridge, New York, Melbourne: Cambridge University Press. ISBN 0-521-47465-5
- Mourtzis et al. 2014** Mourtzis, Dimitris; Doukas, Michael, 2014. Design and Planning of Manufacturing Networks for Mass Customisation and Personalisation: Challenges and Outlook. *Procedia CIRP* **19**, pp. 1-13  
DOI: 10.1016/j.procir.2014.05.004

- Mouy et al. 2015** Mouy, Xavier; Mouy, Pierre-Alain; Hannay, David; Dakin, Tom, 2015.  
JMesh -- A Scalable Web-Based Platform for Visualization and Mining of Passive Acoustic Data.  
In: Cui, Peng (Ed.): *15th IEEE International Conference on Data Mining workshop*, Atlantic City, NJ, USA, 14.11.2015-17.11.2015, pp. 773-779  
ISBN 978-1-4673-8493-3  
DOI: 10.1109/ICDMW.2015.193
- Nadareishvili 2016** Nadareishvili, Irakli, 2016.  
*Microservice architecture: Aligning principles, practices, and culture*.  
Sebastopol, CA: O'Reilly Media.  
ISBN 978-1491956250
- Narkhede et al. 2017** Narkhede, Neha; Shapira, Gwen; Palino, Todd, 2017.  
*Kafka: The definitive guide : real-time data and stream processing at scale*.  
1st edition.  
Sebastopol: O'Reilly Media.  
ISBN 978-1-491-99065-0
- Nemati et al. 2002** Nemati, Hamid R; Steiger, David M; Iyer, Lakshmi S.; Herschel, Richard T., 2002. Knowledge warehouse: an architectural integration of knowledge management, decision support, artificial intelligence and data warehousing.  
*Decision Support Systems* **33** (2), pp. 143-161  
DOI: 10.1016/S0167-9236(01)00141-5
- Neuman 1994** Neuman, B. Clifford, 1994.  
Scale in Distributed Systems.  
In: Casavant, Thomas L; Singhal, Mukesh (Eds.):  
*Readings in distributed computing systems*.  
Los Alamitos, CA, USA: IEEE Computer Society Press, pp. 463-489  
ISBN 978-0-8186-3032-3
- Newman 2015** Newman, Sam, 2015.  
*Building Microservices*.  
1st edition.  
Sebastopol: O'Reilly.  
ISBN 1-4919-5035-8

- Nti et al. 2021** Nti, Isaac Kofi; Nyarko-Boateng, Owusu; Adekoya, Felix Adebayo; Weyori, Benjamin Asubam, 2021. An empirical assessment of different kernel functions on the performance of support vector machines. *Bulletin of Electrical Engineering and Informatics* **10** (6), pp. 3403-3411  
DOI: 10.11591/eei.v10i6.3046
- Nugroho et al. 2008** Nugroho, Ariadi; Chaudron, Michel R.V., 2008. A survey into the rigor of UML use and its perceived impact on quality and productivity. In: Rombach, Dieter; Elbaum, Sebastian; Münch, Jürgen (Eds.): *ESEM'08*, Kaiserslautern, Germany, 09.10.2008-10.10.2008, p. 90  
ISBN 9781595939715  
DOI: 10.1145/1414004.1414020
- Nyhuis et al. 2009** Nyhuis, Peter; Wiendahl, Hans-Peter, 2009. *Fundamentals of Production Logistics*. Berlin, Heidelberg: Springer.  
ISBN 978-3-540-34210-6  
DOI: 10.1007/978-3-540-34211-3
- Obeidat et al. 2015** Obeidat, Muhammad; North, Max; Richardson, Ronny; Rattanak, Vebol; North, Sarah, 2015. Business Intelligence Technology, Applications, and Trends. *International Management Review Journal* **11** (2), pp. 47-56
- OpenFog Consortium 2017** OpenFog Consortium, 2017. *OpenFog Reference Architecture for Fog Computing*. From: [https://www.iiconsortium.org/pdf/OpenFog\\_Reference\\_Architecture\\_2\\_09\\_17.pdf](https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf)  
Viewed: 01.03.2022
- Oussous et al. 2018** Oussous, Ahmed; Benjelloun, Fatima-Zahra; Ait Lahcen, Ayoub; Belfkih, Samir, 2018. Big Data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences* **30** (4), pp. 431-448  
DOI: 10.1016/j.jksuci.2017.06.001

- PAC 2014** PAC, 2014.  
*Predictive Analytics - Mehrwerte, Einsatzbeispiele und Herausforderungen.*  
From:  
[https://www.sas.com/content/dam/SAS/bp\\_de/doc/whitepaper1/ba-wp-pac-predictive-analytics-2308002.pdf](https://www.sas.com/content/dam/SAS/bp_de/doc/whitepaper1/ba-wp-pac-predictive-analytics-2308002.pdf)  
Viewed: 05.04.2021
- Pan et al. 2010** Pan, Sinno Jialin; Yang, Qiang, 2010. A Survey on Transfer Learning.  
*IEEE Transactions on Knowledge and Data Engineering* **22** (10), pp. 1345-1359  
DOI: 10.1109/TKDE.2009.191
- Parnas 1972a** Parnas, David L., 1972.  
Information Distribution Aspects of Design Methodology.  
In: *Information Processing 71. Proceedings of IFIP Congress 71.*  
Amsterdam: North-Holland Publishing Co, pp. 339-344  
ISBN 0-7204-2063-6
- Parnas 1972b** Parnas, David L., 1972. On the criteria to be used in decomposing systems into modules.  
*Communications of the ACM* **15** (12), pp. 1053-1058  
DOI: 10.1145/361598.361623
- Pefferers et al. 2007** Pefferers, Ken; Tuunanen, Tuure; Rothenberger, Marcus A; Chatterjee, Samir, 2007. A Design Science Research Methodology for Information Systems Research.  
*Journal of Management Information Systems* **24** (3), pp. 45-77  
DOI: 10.2753/MIS0742-1222240302
- Pfleeger et al. 2010** Pfleeger, Shari L; Atlee, Joanne M., 2010.  
*Software Engineering: Theory and Practice.*  
4th edition.  
Indien: Prentice Hall.  
ISBN 9780136061694

- Piller 2008** Piller, Frank T., 2008.  
*Mass Customization: Ein wettbewerbsstrategisches Konzept im Informationszeitalter.*  
4th edition.  
Wiesbaden: Dt. Univ.-Verl.  
Gabler Edition Wissenschaft Markt- und Unternehmensentwicklung.  
ISBN 978-3-8350-0355-2
- Poslad 2009** Poslad, Stefan, 2009.  
*Ubiquitous Computing: Smart Devices, Environments and Interactions.*  
2nd edition.  
Hoboken: John Wiley & Sons Ltd.  
ISBN 978-0470035603
- Price et al. 2016** Price, Rosanne; Shanks, Graeme, 2016.  
A Semiotic Information Quality Framework:  
Development and Comparative Analysis.  
In: Willcocks, Leslie P; Sauer, Chris; Lacity, Mary C.  
(Eds.): *Enacting Research Methods in Information Systems.*  
Cham: Springer International Publishing, pp. 219-250  
ISBN 978-3-319-29271-7  
DOI: 10.1007/978-3-319-29272-4\_7
- Priore et al. 2015** Priore, Paolo; Pino, Raúl; Parreño, José; Puente, Javier; Ponte, Borja, 2015. Real-Time Scheduling of Flexible Manufacturing Systems Using Support Vector Machines and Case-Based Reasoning.  
*Journal of Economics, Business and Management* **3** (1), pp. 54-59  
DOI: 10.7763/JOEBM.2015.V3.155
- Przybylek 2010** Przybylek, Adam, 2010.  
An Empirical Assessment of the Impact of Aspect-oriented Programming on Software Modularity.  
In: Loucopoulos, Pericles (Ed.): *Proceedings of the Fifth International Conference on Evaluation of Novel Approaches to Software Engineering*, University of Piraeus, Greece, 22.07.2010-24.07.2010, pp. 139-148  
ISBN 978-989-8425-21-8  
DOI: 10.5220/0003000801390148



- Rajamani et al. 2021** Rajamani, Praveen Kannan; Ageyeva, Tatyana; Kovács, József Gábor, 2021. Personalized Mass Production by Hybridization of Additive Manufacturing and Injection Molding. *Polymers* **13** (2)  
DOI: 10.3390/polym13020309
- Reuter et al. 2017** Reuter, Christina; Brambring, Felix; Hempel, Thomas; Kopp, Phil, 2017. Benefit Oriented Production Data Acquisition for the Production Planning and Control. *Procedia CIRP* **61**, pp. 487-492  
DOI: 10.1016/j.procir.2016.11.142
- Reuter et al. 2016** Reuter, Christina; Brambring, Felix; Weirich, Jan; Kleines, Arne, 2016. Improving Data Consistency in Production Control by Adaptation of Data Mining Algorithms. In: Li, Y. e.a. (Ed.): *9th International Conference on Digital Enterprise Technology (DET 2016): Intelligent Manufacturing in the Knowledge Economy Era*, Nanjing, China, 29.03.2016-31.03.2016, pp. 545-550  
DOI: 10.1016/j.procir.2016.10.107
- Ridge 2015** Ridge, Enda, 2015. *Guerrilla analytics: A practical approach to working with data*. Waltham, MA, USA: Morgan Kaufman. ISBN 978-0128002186
- Rollings et al. 2017** Rollings, Mike; Oestreich, Thomas W., 2017. *The Fundamentals of AI Success for Data and Analytics Leaders*.  
From: <https://www.gartner.com/en/documents/3792883>  
Viewed: 01.05.2022
- Rotem-Gal-Oz 2008** Rotem-Gal-Oz, Arnon, 2008. *Fallacies of Distributed Computing Explained*.  
From: [https://www.researchgate.net/publication/322500050\\_Fallacies\\_of\\_Distributed\\_Computing\\_Explained/references](https://www.researchgate.net/publication/322500050_Fallacies_of_Distributed_Computing_Explained/references)  
Viewed: 01.05.2022

- Rudolf et al. 2015** Rudolf, Stefan; Schrey, Elisabeth, 2015. Datendurchgängigkeit – der Befähiger zur Kollaboration. *Complexity Management Journal* **2015** (02), pp. 13-16
- Runkler 2016** Runkler, Thomas A., 2016. *Data Analytics*. Wiesbaden: Springer Fachmedien. ISBN 978-3-658-14074-8 DOI: 10.1007/978-3-658-14075-5
- Samuel 1959** Samuel, Arthur L., 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development* **3** (3), pp. 210-229 DOI: 10.1147/rd.33.0210
- Savas et al. 2019** Savas, Caner; Dovis, Fabio, 2019. The Impact of Different Kernel Functions on the Performance of Scintillation Detection Based on Support Vector Machines. *Sensors (Basel, Switzerland)* **19** (23), pp. 1-16 DOI: 10.3390/s19235219
- Sayad 2011** Sayad, Saed, 2011. *Real time data mining: The future is here*. Cambridge Ont.: Self-Help Publ. ISBN 978-0986606045
- Schatz et al. 2015** Schatz, Anja; Bauernhansl, Thomas, 2015. Geschäftsmodell-Innovationen: Profitabler wirtschaften mit hohem Vernetzungsgrad. In: Vogel-Heuser, Birgit; Bauernhansl, Thomas; Hompel, Michael ten (Eds.): *Handbuch Industrie 4.0: Profitabler wirtschaften mit hohem Vernetzungsgrad*. Wiesbaden: Springer Fachmedien, pp. 1-15 ISBN 978-3-662-45537-1 DOI: 10.1007/978-3-662-45537-1\_95-1
- Schilit et al. 1994** Schilit, Bill; Adams, Norman; Want, Roy, 1994. Context-Aware Computing Applications. In: *1994 First Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, USA, 08.12.1994-09.12.1994, pp. 85-90 ISBN 978-0-7695-3451-0 DOI: 10.1109/WMCSA.1994.16

- 
- Schnider et al. 2016** Schnider, Dani; Jordan, Claus; Welker, Peter; Wehner, Joachim, 2016.  
*Data Warehouse Blueprints: Business Intelligence in der Praxis*.  
München: Hanser.  
ISBN 978-3-446-45075-2  
DOI: 10.3139/9783446451117
- Schniederjans 2015** Schniederjans, Marc J., 2015.  
*Business analytics principles, concepts, and applications with SAS: What, why, and how*.  
Upper Saddle River, NJ, USA: Pearson Pub.  
ISBN 978-0-13-398940-3
- Schönsleben 2012** Schönsleben, Paul, 2012. Methods and tools that support a fast and efficient design-to-order process for parameterized product families.  
*CIRP Annals* **61** (1), pp. 179-182  
DOI: 10.1016/j.cirp.2012.03.071
- Schotten 1998** Schotten, Martin, 1998.  
*Produktionsplanung und -steuerung: Grundlagen, Gestaltung und Konzepte*.  
Berlin, Heidelberg: Springer.  
VDI-Buch.  
ISBN 978-3-662-09475-4  
DOI: 10.1007/978-3-662-09474-7
- Schuh et al. 2018** Schuh, Günther; Prote, Jan-Phillip; Luckert, Melanie; Sauer mann, Frederick, 2018. Determination of order specific transition times for improving the adherence to delivery dates by using data mining algorithms.  
*Procedia CIRP* **72**, pp. 169-173  
DOI: 10.1016/j.procir.2018.03.236
- Schuh et al. 2017** Schuh, Günther; Reuter, Christina; Prote, Jan-Philipp; Brambring, Felix; Ays, Julian, 2017.  
Increasing data integrity for improving decision making in production planning and control.  
In: CIRP (Ed.): *CIRP Annals*, pp. 425-428  
DOI: 10.1016/j.cirp.2017.04.003

- scikit-learn developers 2022** scikit-learn developers, 2022.  
*1.4. Support Vector Machines — scikit-learn 1.1.dev0 documentation.*  
From: <https://scikit-learn.org/dev/modules/svm.html>  
Viewed: 12.04.2022
- SCOR 12.0** SCOR 12.0.  
*Supply Chain Operations Reference Model.*
- Seltman 2018** Seltman, Howard J., 2018.  
*Experimental Design and Analysis.*  
From:  
<http://www.stat.cmu.edu/~hseltman/309/Book/Book.pdf>  
Viewed: 25.04.2019
- Shadija et al. 2017** Shadija, Dharmendra; Rezai, Mo; Hill, Richard, 2017.  
Towards an understanding of microservices.  
In: Zhang, Jie (Ed.): *Addressing global challenges through automation and computing.*  
Piscataway, NJ, USA: IEEE, pp. 1-6  
ISBN 978-0-7017-0260-1  
DOI: 10.23919/ConAC.2017.8082018
- Shah et al. 2007** Shah, Rachna; Ward, Peter T., 2007. Defining and developing measures of lean production.  
*Journal of Operations Management* **25** (4), pp. 785-805  
DOI: 10.1016/j.jom.2007.01.019
- Shao et al. 2014** Shao, Guodong; Shin, Seung-Jun; Jain, Sanjay, 2014.  
Data analytics using simulation for smart manufacturing.  
In: *Proceedings of the Winter Simulation Conference 2014*, Savannah, GA, USA, 07.12.2014-10.12.2014, pp. 2192-2203  
ISBN 978-1-4799-7486-3  
DOI: 10.1109/WSC.2014.7020063

- Shin et al. 2016** Shin, Hoo-Chang; Roth, Holger R.; Gao, Mingchen; Le Lu; Xu, Ziyue; Nogues, Isabella; Yao, Jianhua; Mollura, Daniel; Summers, Ronald M., 2016. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE transactions on medical imaging* **35** (5), pp. 1285-1298  
DOI: 10.1109/TMI.2016.2528162
- Slack 1983** Slack, Nigel, 1983. Flexibility as a Manufacturing Objective. *International Journal of Operations & Production Management* **3** (3), pp. 4-13  
DOI: 10.1108/eb054696
- Slack 1987** Slack, Nigel, 1987. The Flexibility of Manufacturing Systems. *International Journal of Operations & Production Management* **7** (4), pp. 35-45  
DOI: 10.1108/eb054798
- Soltanpoor et al. 2016** Soltanpoor, Reza; Sellis, Timos, 2016. Prescriptive Analytics for Big Data. In: Cheema, Muhammad A; Zhang, Wenjie; Chang, Lijun (Eds.): *Databases theory and applications*. Cham: Springer, pp. 245-256  
ISBN 978-3-319-46921-8  
DOI: 10.1007/978-3-319-46922-5\_19
- Stahl et al. 2013** Stahl, Frederic; Gaber, Mohamed Medhat; Bramer, Max, 2013. Scaling up Data Mining Techniques to Large Datasets Using Parallel and Distributed Processing. In: Rausch, Peter; Sheta, Alaa F; Ayes, Aladdin (Eds.): *Business Intelligence and Performance Management*. London: Springer, pp. 243-259  
ISBN 978-1-4471-4865-4  
DOI: 10.1007/978-1-4471-4866-1\_16

- Stein et al. 2019** Stein, Alexander; Zillekens, Marcel; Khan, Marius, 2019.  
A Microservice architecture for monitoring, processing and predicting climate data in animal husbandry.  
In: *Online Proceedings of the 2nd International Conference on Microservices*, Dortmund, Germany, 19.02.2019-21.02.2019
- Stevens et al. 1974** Stevens, Wayne; Myers, Glenford; Constantine, Larry, 1974. Structured design.  
*IBM Systems Journal* **13** (2), pp. 115-139  
DOI: 10.1147/sj.132.0115
- Stonebraker et al. 2018** Stonebraker, Michael; Ilyas, Ihab F., 2018. Data Integration: The Current Status and the Way Forward.  
*Bulletin of the Technical Committee on Data Engineering* **41** (2), pp. 3-9
- Sutton et al. 2018** Sutton, Richard S.; Barto, Andrew, 2018.  
*Reinforcement learning: An introduction*.  
2nd edition.  
Cambridge, MA, London: The MIT Press.  
Adaptive computation and machine learning.  
ISBN 978-0262039246
- Sutton et al. 2005** Sutton, Charles; McCallum, Andrew, 2005.  
Composition of conditional random fields for transfer learning.  
In: Mooney, Raymond J. (Ed.): *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, Vancouver, British Columbia, Canada, 10.6.2005-10.8.2005, pp. 748-754  
DOI: 10.3115/1220575.1220669
- Svahnberg et al. 2002** Svahnberg, Mikael; Wohlin, Claes, 2002.  
Consensus Building when Comparing Software Architectures.  
In: Goos, Gerhard; Hartmanis, Juris; van Leeuwen, Jan; Oivo, Markku; Komi-Sirviö, Seija (Eds.): *Product Focused Software Process Improvement*.  
Berlin, Heidelberg: Springer, pp. 436-452  
ISBN 978-3-540-00234-5  
DOI: 10.1007/3-540-36209-6\_36

- Szczuka et al. 2014** Szczuka, Marcin; Sosnowski, Łukasz; Krasuski, Adam; Kreński, Karol, 2014. Using Domain Knowledge in Initial Stages of KDD: Optimization of Compound Object Processing. *Fundamenta Informaticae* **129** (4), pp. 341-364  
DOI: 10.3233/FI-2014-975
- Taha 2017** Taha, Hamdy A., 2017. *Operations research: An introduction*. 10th edition. Harlow, England: Pearson. ISBN 978-1292165547
- Takai 2017** Takai, Daniel, 2017. *Architektur für Websysteme: Serviceorientierte Architektur, Microservices, domänengetriebener Entwurf*. München: Hanser. ISBN 978-3-446-45056-1  
DOI: 10.3139/9783446452480
- Talia 2013** Talia, Domenico, 2013. Clouds for Scalable Big Data Analytics. *Computer* **46** (5), pp. 98-101  
DOI: 10.1109/MC.2013.162
- Tamura et al. 2015** Tamura, Yasumasa; Iizuka, Hiroyuki; Yamamoto, Masahito; Furukawa, Masashi, 2015. Application of local clustering organization to reactive job-shop scheduling. *Soft Computing* **19** (4), pp. 891-899  
DOI: 10.1007/s00500-014-1416-4
- Tempelmeier 2016** Tempelmeier, Horst, 2016. *Produktionsplanung in Supply Chains*. 4th edition. Norderstedt: Books on Demand. ISBN 978-3741208119
- The Open Group 2009** The Open Group, 2009. *SOA source book*. 1st edition. Zaltbommel: Van Haren Publishing. The Open Group series. ISBN 978-9087535032

- ThoughtWorks 2015** ThoughtWorks, 2015.  
*Inverse Conway Maneuver*.  
From:  
<https://www.thoughtworks.com/de/radar/techniques/inverse-conway-maneuver>  
Viewed: 05.04.2021
- Torrey et al. 2010** Torrey, Lisa; Shavlik, Jude, 2010.  
Transfer Learning.  
In: Olivas, Emilio S; Guerrero, José D.M; Martínez-Sober, Marcelino; Magdalena-Benedito, Jose R; Serrano López, Antonio J. (Eds.): *Handbook of Research on Machine Learning Applications and Trends*: IGI Global, pp. 242-264  
ISBN 9781605667669  
DOI: 10.4018/978-1-60566-766-9.ch011
- Tory et al. 2004** Tory, Melanie; Möller, Torsten, 2004. Human factors in visualization research.  
*IEEE transactions on visualization and computer graphics* **10** (1), pp. 72-84  
DOI: 10.1109/TVCG.2004.1260759
- Tsai et al. 2015** Tsai, Chun-Wei; Lai, Chin-Feng; Chao, Han-Chieh; Vasilakos, Athanasios V., 2015. Big data analytics: a survey.  
*Journal of Big Data* **2** (1), p. 1170  
DOI: 10.1186/s40537-015-0030-3
- Ulrich et al. 1976** Ulrich, Peter; Hill, Wilhelm, 1976.  
Wissenschaftstheoretische Grundlagen der Betriebswirtschaftslehre (Teil I).  
*Wirtschaftswissenschaftliches Studium : WiSt : Zeitschrift für Studium und Forschung* **5** (7), pp. 304-309
- van der Lans 2012** van der Lans, Rick F., 2012.  
*Data virtualization for business intelligence architectures: Revolutionizing data integration for data warehouses*.  
Amsterdam, Boston: Elsevier/MK.  
The Morgan Kaufmann Series on Business Intelligence.  
ISBN 978-0123944252



- van Steen et al. 2017** van Steen, Maarten; Tanenbaum, Andrew S., 2017. *Distributed systems*. 3rd edition. London: Pearson Education. ISBN 978-1543057386
- Vavouras et al. 1999** Vavouras, Athanasios; Gatzui, Stella; Dittrich, Klaus R., 1999. The SIRIUS Approach for Refreshing Data Warehouses Incrementally. In: Buchmann, Alejandro P. (Ed.): *Datenbanksysteme in Büro, Technik und Wissenschaft*. Berlin, Heidelberg: Springer, pp. 80-96 ISBN 978-3-540-65606-7 DOI: 10.1007/978-3-642-60119-4\_6
- VDI 5600** VDI 5600 Blatt 1:2007. *Fertigungsmanagementsysteme*.
- Vernon 2016** Vernon, Vaughn, 2016. *Domain-driven design distilled*. Boston: Addison-Wesley. ISBN 978-0-13-443442-1
- Walia et al. 2020** Walia, Neha; Kalia, Arvind, 2020. Tools in Data Mining A Comparative Analysis. *International Journal of New Innovations in Engineering and Technology* **12** (4), pp. 89-94
- Wand et al. 1996** Wand, Yair; Wang, Richard Y., 1996. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM* **39** (11), pp. 86-95 DOI: 10.1145/240455.240479
- Wang et al. 1996** Wang, Richard Y.; Strong, Diane M., 1996. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems* **12** (4), pp. 5-33 DOI: 10.1080/07421222.1996.11518099

- Wang et al. 2016** Wang, Gang; Gunasekaran, Angappa; Ngai, Eric W.T.; Papadopoulos, Thanos, 2016. Big data analytics in logistics and supply chain management: Certain investigations for research and applications. *International Journal of Production Economics* **176**, pp. 98-110  
DOI: 10.1016/j.ijpe.2016.03.014
- Wang et al. 2019** Wang, Tianyang; Huan, Jun; Zhu, Michelle, 2019. Instance-Based Deep Transfer Learning. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa Village, HI, USA, 07.01.2019 - 11.01.2019, pp. 367-375  
ISBN 978-1-7281-1975-5  
DOI: 10.1109/WACV.2019.00045
- Wehner et al. 2016** Wehner, Daniel; Dangelmaier, Manfred; Hampel, Martina; Paulus-Rohmer, Dominik; Hörcher, Günter; Krieg, Sabine; Rüger, Marc; Demont, Anja; Held, Michael; Ilg, Robert, 2016. *Mass Personalization – Mit personalisierten Produkten zum Business-to-User (B2U)*.  
From:  
<https://www.ipa.fraunhofer.de/de/Publikationen/studien/mass-personalization.html>  
Viewed: 01.11.2023
- Westkämper et al. 2000** Westkämper, Engelbert; Schmidt, Thomas; Wiendahl, Hans-Hermann, 2000. Production Planning and Control with Learning Technologies Simulation and Optimization of Complex Production Processes. In: *Knowledge-Based Systems*. London: Elsevier, pp. 839-887  
ISBN 978-0-12-443875-0  
DOI: 10.1016/B978-012443875-0/50029-X
- Westkämper et al. 2006** Westkämper, Engelbert; Decker, Markus, 2006. *Einführung in die Organisation der Produktion*. Berlin, Heidelberg: Springer. Springer-Lehrbuch.  
ISBN 9783540260394  
DOI: 10.1007/3-540-30764-8

- Wierse et al. 2017** Wierse, Andreas; Riedel, Till, 2017. *Smart Data Analytics: Zusammenhänge erkennen, Potentiale nutzen, Big Data verstehen*. ISBN 978-3110461848  
DOI: 10.1515/9783110463958
- Wilde et al. 2007** Wilde, Thomas; Hess, Thomas, 2007. Forschungsmethoden der Wirtschaftsinformatik. *Wirtschaftsinformatik* **49** (4), pp. 280-287  
DOI: 10.1007/s11576-007-0064-z
- Winter et al. 2010** Winter, Robert; Baskerville, Richard, 2010. Methodik der Wirtschaftsinformatik. *Wirtschaftsinformatik* **52** (5), pp. 257-258  
DOI: 10.1007/s11576-010-0242-2
- Witten et al. 2017** Witten, Ian H.; Pal, Christopher J.; Frank, Eibe; Hall, Mark A., 2017. *Data mining: Practical machine learning tools and techniques*. 4th edition. Cambridge, MA: Morgan Kaufmann. ISBN 978-0128042915
- WKWI 2011** WKWI, 2011. *Profil der Wirtschaftsinformatik*. From: <https://wi-lex.de/index.php/lexikon/uebergreifender-teil/disziplinen-der-wi/wirtschaftsinformatik/profil-der-wirtschaftsinformatik/>  
Viewed: 01.11.2023
- Wolff 2018** Wolff, Eberhard, 2018. *Microservices: Grundlagen flexibler Softwarearchitekturen*. 2nd edition. Heidelberg: dpunkt. ISBN 978-3864905551
- Wong et al. 2004** Wong, Pak Chung; Thomas, J., 2004. Visual Analytics. *IEEE Computer Graphics and Applications* **24** (5), pp. 20-21  
DOI: 10.1109/MCG.2004.39

- Woo et al. 2018** Woo, Jungyub; Shin, Seung-Jun; Seo, Wonchul; Meilanitasari, Prita, 2018. Developing a big data analytics platform for manufacturing systems: architecture, method, and implementation. *The International Journal of Advanced Manufacturing Technology* **99** (9-12), pp. 2193-2217  
DOI: 10.1007/s00170-018-2416-9
- Yosinski et al. 2014** Yosinski, Jason; Clune, Jeff; Bengio, Yoshua; Lipson, Hod, 2014. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems* **27**, pp. 3320-3328  
DOI: 10.5555/2969033.2969197
- Yourdon et al. 1979** Yourdon, Edward; Constantine, Larry L., 1979. *Structured design: Fundamentals of a discipline of computer program and systems design*. Englewood Cliffs, NJ, USA: Prentice-Hall.  
ISBN 0138544719
- Yousefpour et al. 2019a** Yousefpour, Ashkan; Fung, Caleb; Nguyen, Tam; Kadiyala, Krishna; Jalali, Fatemeh; Niakanlahiji, Amirreza; Kong, Jian; Jue, Jason P., 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture* **98**, pp. 289-330  
DOI: 10.1016/j.sysarc.2019.02.009
- Yousefpour et al. 2019b** Yousefpour, Ashkan; Devic, Siddhartha; Nguyen, Brian Q; Kreidieh, Aboudy; Liao, Alan; Bayen, Alexandre M; Jue, Jason P., 2019. Guardians of the Deep Fog. In: *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, New York, NY, USA, 10.11.2019-13.11.2019, pp. 25-31  
ISBN 9781450370134  
DOI: 10.1145/3363347.3363366

- 
- Zangemeister 2014** Zangemeister, Christof, 2014. *Nutzwertanalyse in der Systemtechnik: Eine Methodik zur multidimensionalen Bewertung und Auswahl von Projektalternativen*. 5th edition. Norderstedt: Books on Demand. ISBN 9783923264001
- Zhang 2013** Zhang, Rui, 2013. A Simulated Annealing-Based Heuristic Algorithm for Job Shop Scheduling to Minimize Lateness. *International Journal of Advanced Robotic Systems* **10** (4), p. 214  
DOI: 10.5772/55956
- Zhou et al. 2019** Zhou, Xiang; Peng, Xin; Xie, Tao; Sun, Jun; Ji, Chao; Li, Wenhai; Ding, Dan, 2019. Delta Debugging Microservice Systems with Parallel Optimization. *IEEE Transactions on Services Computing* **15** (1), 16–29  
DOI: 10.1109/TSC.2019.2919823
- Zhu et al. 2021** Zhu, Haoyu; Woo, Jong Hun, 2021. Hybrid NHPSO-JTVAC-SVM Model to Predict Production Lead Time. *Applied Sciences* **11** (14), pp. 1-16  
DOI: 10.3390/app11146369



# 10 Appendix

## 10.1 Appendix A

Complete component diagram of the proposed solution illustrating the main interfaces.

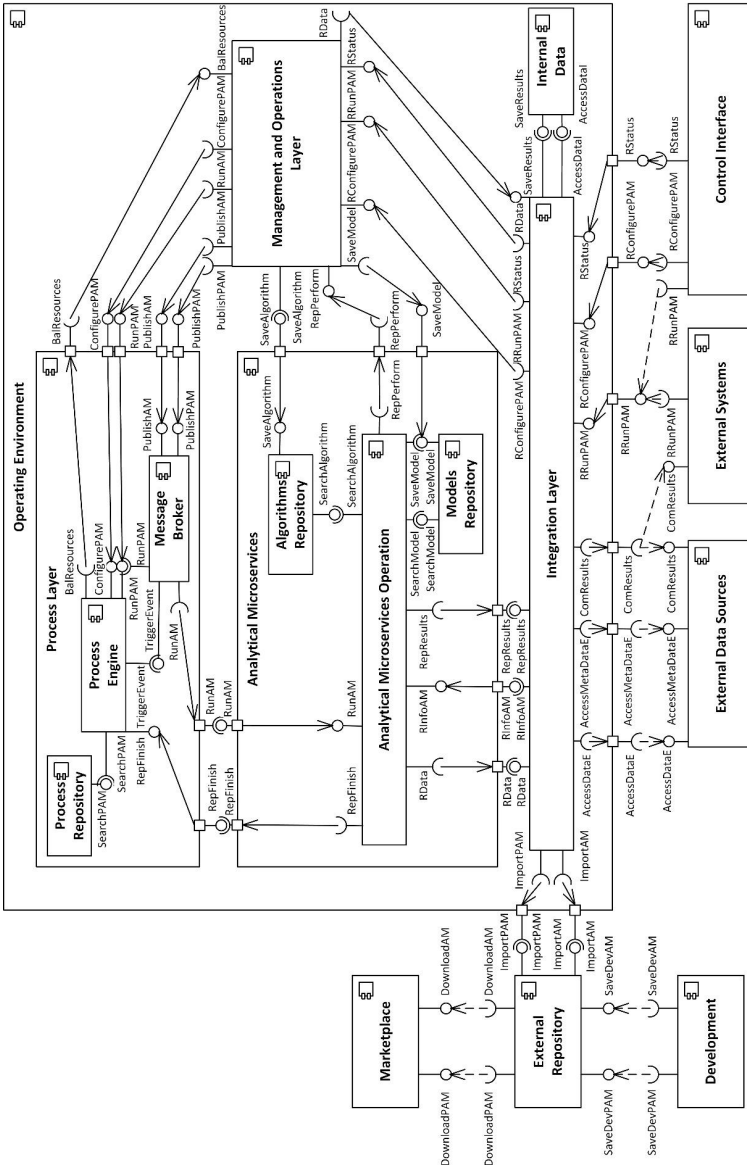


Figure 10-1: Detailed component diagram of the solution



## 10.2 Appendix B

Detailed description of factors and weights for the evaluation of application scenarios of the proposed solution (and comparison with alternative approaches).

**Table 10-1: Factors and weights for the evaluation of application scenarios**

Scenario	Solution	W <sub>1</sub>	DU	W <sub>2</sub>	FU	W <sub>3</sub>	GI	W <sub>4</sub>	RC	Total Benefit	Total Cost	Utility
E1 – F1	Business Intelligence Solutions	0.7	4	0.5	2	1	2	1	1	3.80	3.00	1.27
E1 – F1	Data Mining Suites	0.7	4	0.5	4	1	3	1	1	4.80	4.00	1.20
E1 – F1	Tailored Solutions	0.7	5	0.5	1	1	2	1	1	4.00	3.00	1.33
E1 – F1	Analytics as a Service	0.7	4	0.5	4	1	2	1	2	4.80	4.00	1.20
E1 – F1	Analytical Microservices	0.7	5	0.5	5	1	3	1	2	6.00	5.00	1.20
E1 – F2	Business Intelligence Solutions	0.7	3	0.5	2	1	2	1	1	3.10	3.00	1.03
E1 – F2	Data Mining Suites	0.7	4	0.5	4	1	3	1	1	4.80	4.00	1.20
E1 – F2	Tailored Solutions	0.7	5	0.5	1	1	2	1	1	4.00	3.00	1.33
E1 – F2	Analytics as a Service	0.7	4	0.5	4	1	2	1	2	4.80	4.00	1.20
E1 – F2	Analytical Microservices	0.7	5	0.5	5	1	3	1	2	6.00	5.00	1.20
E1 – F3	Business Intelligence Solutions	0.8	2	0.5	2	1	3	1	1	2.60	4.00	0.65
E1 – F3	Data Mining Suites	0.8	3	0.5	3	1	3	1	1	3.90	4.00	0.98
E1 – F3	Tailored Solutions	0.8	5	0.5	1	1	3	1	1	4.50	4.00	1.13
E1 – F3	Analytics as a Service	0.8	3	0.5	4	1	2	1	2	4.40	4.00	1.10
E1 – F3	Analytical Microservices	0.8	5	0.5	5	1	3	1	2	6.50	5.00	1.30
E1 – F4	Business Intelligence Solutions	0.8	2	0.5	2	1	3	1	1	2.60	4.00	0.65
E1 – F4	Data Mining Suites	0.8	3	0.5	3	1	3	1	1	3.90	4.00	0.98
E1 – F4	Tailored Solutions	0.8	5	0.5	1	1	4	1	1	4.50	5.00	0.90
E1 – F4	Analytics as a Service	0.8	3	0.5	4	1	2	1	2	4.40	4.00	1.10
E1 – F4	Analytical Microservices	0.8	5	0.5	5	1	3	1	2	6.50	5.00	1.30

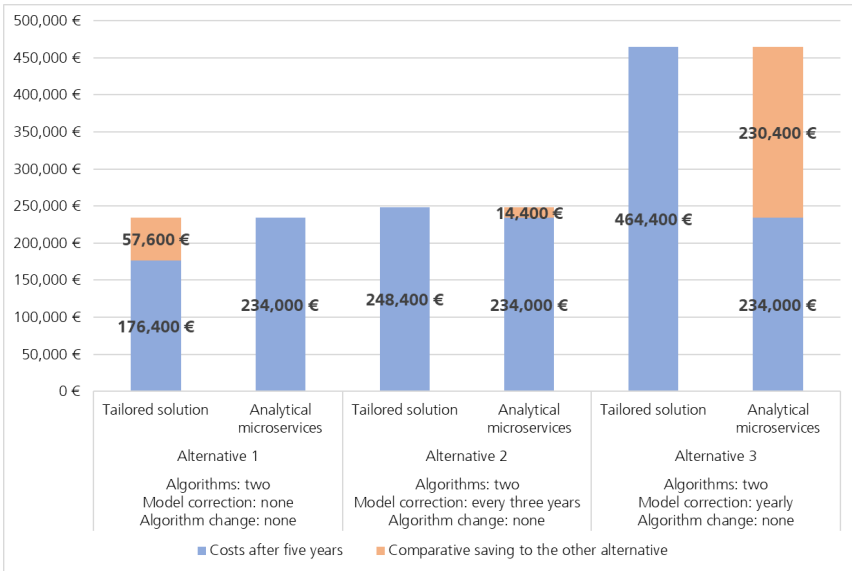
Scenario	Solution	W <sub>1</sub>	DU	W <sub>2</sub>	FU	W <sub>3</sub>	GI	W <sub>4</sub>	RC	Total Benefit	Total Cost	Utility
E2 – F1	Business Intelligence Solutions	0.8	3	0.8	2	1	3	1	2	4.00	5.00	0.80
E2 – F1	Data Mining Suites	0.8	4	0.8	4	1	3	1	2	6.40	5.00	1.28
E2 – F1	Tailored Solutions	0.8	5	0.8	1	1	3	1	1	4.80	4.00	1.20
E2 – F1	Analytics as a Service	0.8	4	0.8	4	1	2	1	2	6.40	4.00	1.60
E2 – F1	Analytical Microservices	0.8	5	0.8	5	1	3	1	3	8.00	6.00	1.33
E2 – F2	Business Intelligence Solutions	0.8	2	0.8	1	1	3	1	2	2.40	5.00	0.48
E2 – F2	Data Mining Suites	0.8	3	0.8	3	1	3	1	2	4.80	5.00	0.96
E2 – F2	Tailored Solutions	0.8	5	0.8	1	1	4	1	1	4.80	5.00	0.96
E2 – F2	Analytics as a Service	0.8	3	0.8	3	1	2	1	2	4.80	4.00	1.20
E2 – F2	Analytical Microservices	0.8	5	0.8	5	1	3	1	3	8.00	6.00	1.33
E2 – F3	Business Intelligence Solutions	1	2	0.8	1	1	3	1	2	2.80	5.00	0.56
E2 – F3	Data Mining Suites	1	3	0.8	2	1	3	1	2	4.60	5.00	0.92
E2 – F3	Tailored Solutions	1	5	0.8	1	1	4	1	1	5.80	5.00	1.16
E2 – F3	Analytics as a Service	1	3	0.8	3	1	3	1	2	5.40	5.00	1.08
E2 – F3	Analytical Microservices	1	5	0.8	5	1	3	1	3	9.00	6.00	1.50
E2 – F4	Business Intelligence Solutions	1	1	0.8	1	1	3	1	3	1.80	6.00	0.30
E2 – F4	Data Mining Suites	1	3	0.8	2	1	3	1	2	4.60	5.00	0.92
E2 – F4	Tailored Solutions	1	5	0.8	1	1	4	1	1	5.80	5.00	1.16
E2 – F4	Analytics as a Service	1	3	0.8	2	1	3	1	2	4.60	5.00	0.92
E2 – F4	Analytical Microservices	1	5	0.8	5	1	3	1	3	9.00	6.00	1.50
E3 – F1	Business Intelligence Solutions	1	2	1	2	1	3	1	2	4.00	5.00	0.80
E3 – F1	Data Mining Suites	1	4	1	3	1	3	1	2	7.00	5.00	1.40
E3 – F1	Tailored Solutions	1	5	1	1	1	4	1	1	6.00	5.00	1.20
E3 – F1	Analytics as a Service	1	4	1	3	1	3	1	2	7.00	5.00	1.40
E3 – F1	Analytical Microservices	1	5	1	5	1	3	1	3	10.00	6.00	1.67

Scenario	Solution	W <sub>1</sub>	DU	W <sub>2</sub>	FU	W <sub>3</sub>	GI	W <sub>4</sub>	RC	Total Benefit	Total Cost	Utility
E3 – F2	Business Intelligence Solutions	1	2	1	1	1	3	1	2	3.00	5.00	0.60
E3 – F2	Data Mining Suites	1	3	1	3	1	3	1	2	6.00	5.00	1.20
E3 – F2	Tailored Solutions	1	5	1	1	1	4	1	1	6.00	5.00	1.20
E3 – F2	Analytics as a Service	1	3	1	3	1	3	1	2	6.00	5.00	1.20
E3 – F2	Analytical Microservices	1	5	1	5	1	3	1	3	10.00	6.00	1.67
E3 – F3	Business Intelligence Solutions	1	1	1	1	1	5	1	2	2.00	7.00	0.29
E3 – F3	Data Mining Suites	1	2	1	2	1	4	1	2	4.00	6.00	0.67
E3 – F3	Tailored Solutions	1	5	1	1	1	5	1	1	6.00	6.00	1.00
E3 – F3	Analytics as a Service	1	2	1	2	1	3	1	2	4.00	5.00	0.80
E3 – F3	Analytical Microservices	1	5	1	5	1	3	1	3	10.00	6.00	1.67
E3 – F4	Business Intelligence Solutions	1	1	1	1	1	5	1	2	2.00	7.00	0.29
E3 – F4	Data Mining Suites	1	2	1	2	1	4	1	2	4.00	6.00	0.67
E3 – F4	Tailored Solutions	1	5	1	1	1	5	1	1	6.00	6.00	1.00
E3 – F4	Analytics as a Service	1	2	1	2	1	3	1	2	4.00	5.00	0.80
E3 – F4	Analytical Microservices	1	5	1	5	1	3	1	3	10.00	6.00	1.67

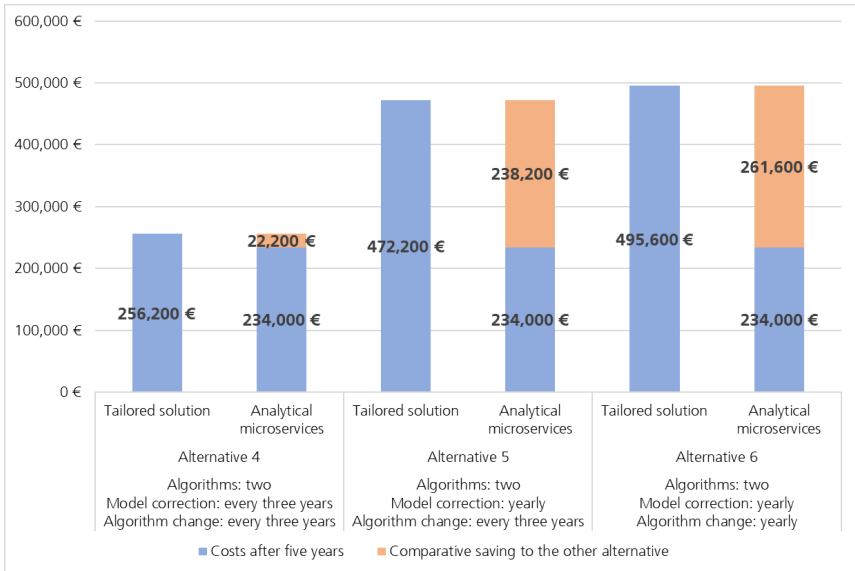
DU: Direct Utilization; FU: Further Use; GI: General Investments; RC: Running Costs; W<sub>1</sub>: Weight Direct Utilization; W<sub>2</sub>: Weight Further Use; W<sub>3</sub>: Weight General Investments; W<sub>4</sub>: Weight Running Costs

### 10.3 Appendix C

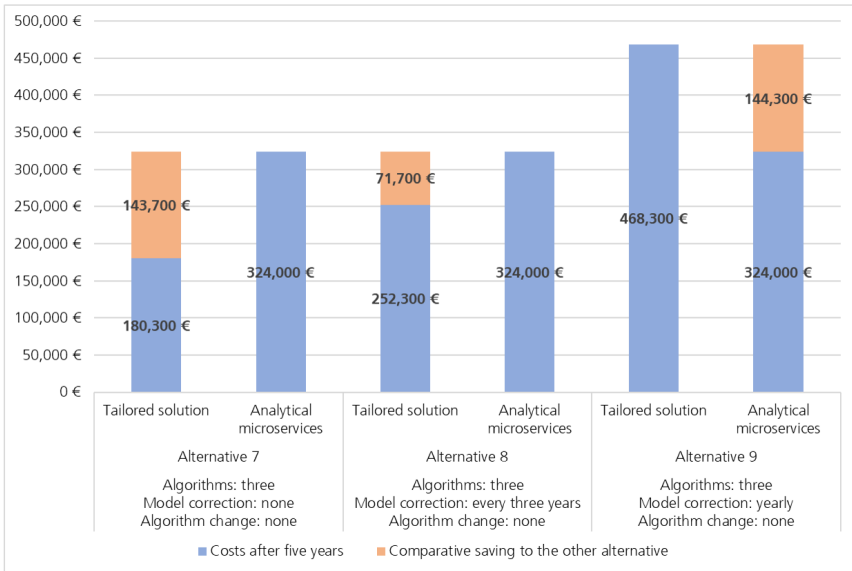
Evaluation of alternatives 1 to 12 of the industrial use case with ten additional models instead of seven.



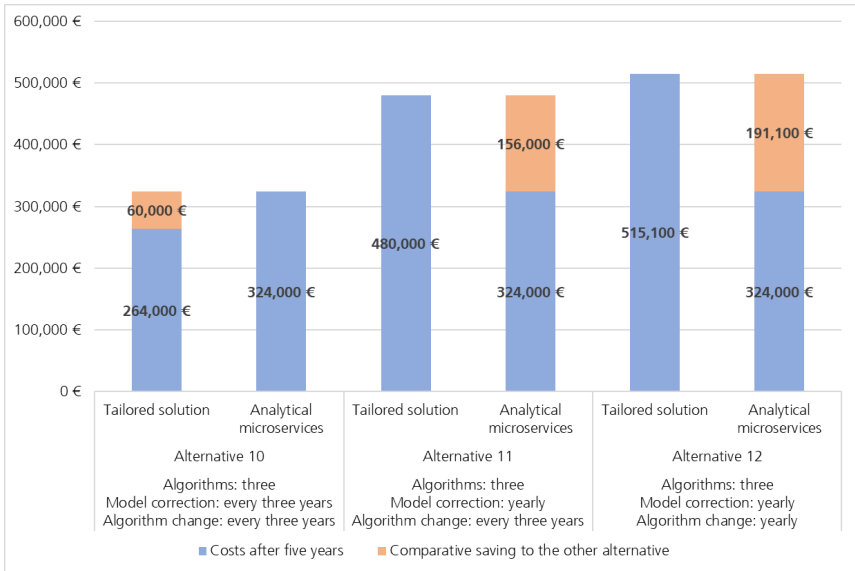
**Figure 10-2: Comparison of cash flows of alternatives 1 to 3 (ten models)**



**Figure 10-3: Comparison of cash flows of alternatives 4 to 6 (ten models)**



**Figure 10-4: Comparison of cash flows of alternatives 7 to 9 (ten models)**



**Figure 10-5: Comparison of cash flows of alternatives 10 to 12 (ten models)**







