Today’s computer networks, like the internet or intranets of companies, interconnect a large number of devices as well as persons. Within such networks, content can be exchanged and services can be utilised. The more complex such networks become, the more opportunities might exist to choose a desired service or content. If multiple of those choices need to be made periodically or in short time, manual selection may not be a suitable procedure. From this fact the question arises, how the service or content selection process can be managed efficiently.

In this book, the author Rico Kusber investigates scientifically the process of selecting services or content for deployment on networked computing systems. He presents an algorithmic approach how services and content can be selected automatically in accordance with user preferences and needs. The book addresses scientists and developers in the fields of Autonomic Computing, Ubiquitous Computing, as well as Service and Content Management.

Rico Kusber, Dipl.-Inf.: Researcher at the University of Kassel, Chair for Communication Technology (ComTec); PhD in Computer Science at the University of Kassel in 2010; Degree in Computer Science at Chemnitz University of Technology in 2006, specialisation in Artificial Intelligence

ISBN 978-3-86219-128-4

Rico Kusber
An algorithmic approach to service and content deployment decision making
Rico Kusber

An algorithmic approach to service and content deployment decision making
Abstract

Today’s computer networks, like the internet or intranets of companies, interconnect a large number of devices as well as persons. Within such networks, content can be exchanged and services can be utilised. The more complex such networks become, the more opportunities might exist to choose a desired service or content. If multiple of those choices need to be made periodically or in short time, manual selection may not be a suitable procedure. From this fact the question arises, how the service or content selection process can be managed efficiently.

That question needs to be answered for various research and application domains. Consider, for example, users who frequently want to obtain multimedia content without choosing manually one out of many alternatives where to get the desired data from. Pervasively available devices in Ubiquitous Computing scenarios might have very limited computational capabilities. For that reason it may be necessary for them to retrieve content on demand or utilise services remotely from time to time, which requires an efficient content or service selection mechanism. Autonomic Computing systems like computing clouds, that work without human administration, need to decide upon the utilisation of services and resources in a completely user independent, though efficient manner.

In this thesis we investigate the process of selecting services or content for deployment on networked computing systems. We analyse which information is necessary and which is available, and how this information can be used to come to meaningful deployment decisions. After exploring the given question theoretically, we present an approach we developed, implemented, and evaluated in order to solve the concerned issue. According to the presented approach, we define input and output interfaces, and use them to circumscribe the boundaries of our research. Furthermore, we utilise experiments to illustrate the limits and possibilities of applying our approach. Based on our research results, arising challenges are described.

The work we present contributes to theory and practice of research in the fields of network communication technology, autonomic computing, and related areas because we provide a property for selecting services or content automatically and meaningfully. We created facilities for controlling the concerned algorithm and for improving its computational performance. Moreover, we developed one particular prototype
implementation, application scenarios, and example parameter sets. Investigations we performed theoretically are complemented and confirmed by experiments and simulations.
Zusammenfassung


Acknowledgement

It is a pleasure to thank the people who contributed to this work and supported me along my way of writing this thesis.

First of all I am very grateful to my supervisor, Prof. Dr. Klaus David, for giving me the opportunity to carry out this work, for guiding and directing my scientific way, for plenty of reflective discussions, and for his constant encouragement.

I want to thank very much Dr. James Irvine for accepting to assess this work as second supervisor, as well as Prof. Dr. Albert Zündorf and Prof. Dr. Dieter Wloka for being members of the PhD committee.

Moreover, I wish to thank the following people: Prof. Dr. Werner Dilger for sparking off my enthusiasm for Artificial Intelligence; Dr. Sandra Haseloff and Dr. Olaf Drögehorn for inspiring discussions, pushing me to my first publications on deployment decision making, and expressing confidence in my work; the former and current team at ComTec for exchanging thoughts, mastering the many tasks along and besides working on a PhD, and having a great time in Kassel; Dr. Stephan Sigg for assessing my work and pointing out what I have missed; Tom Norberg for the challenging attempt to improve my capability of expressing myself in English; Dr. Björn Wüst for elaborating first ideas on deployment decision making; Nermin Brgulja for bearing all my peculiarities, and sharing thoughts about the world; the many students who worked with me, among them Daniel Gartz, Stefan Reiser, and Sebastian Böttger.

I would like to acknowledge the European Commission for funding the Integrated Project CASCADAS “Component-ware for Autonomic, Situation-aware Communications, And Dynamically Adaptable Services” (FET Proactive Initiative, IST-2004-2.3.4 Situated and Autonomic Communications) within the 6th IST Framework Program. I also wish to thank other project partners for the fruitful cooperation within the project.

I thank so much all my friends for their support, encouragement whenever needed, and their friendship after all. Ich danke mich bei Falko Weiß, der mich auf meinen akademischen Weg geführt hat und mich jederzeit bei meinen Vorhaben unterstützt. Schließlich möchte ich mich bei meiner Familie, besonders meinen Eltern, meiner Schwester und meiner Freundin Christiane, bedanken. Danke für die Unterstützung, Aufmunterung und Geduld! Danke für die Opfer, die Ihr meinetwegen gebracht habt!
Für meine Familie.
# Table of Contents

Abbreviations and Notation.......................................................................................... xv
Publications.................................................................................................................. xxi

1 Introduction .............................................................................................................. 23
   1.1 Problem statement ............................................................................................. 23
      1.1.1 The context aware autonomic advertisement scenario............................ 23
      1.1.2 The music video scenario ........................................................................... 25
      1.1.3 Problem and demands ................................................................................ 27
   1.2 Methodology and scope ..................................................................................... 29
   1.3 Contribution ....................................................................................................... 31
   1.4 Summary and structure of this thesis .................................................................. 33

2 Related work and fundamentals.................................................................................. 35
   2.1 Related research topics ...................................................................................... 35
      2.1.1 Starting with artificial intelligence ............................................................. 35
      2.1.2 Ubiquitous computers pervading daily lives ................................................. 36
      2.1.3 Handling complexity with autonomic computing ...................................... 36
      2.1.4 Multiple entities in service oriented architectures ....................................... 39
      2.1.5 Web services as exemplary technology ...................................................... 40
      2.1.6 Ensuring quality when invoking services .................................................... 41
      2.1.7 Discovering services and content ............................................................... 42
      2.1.8 Describing services and content ................................................................. 43
      2.1.9 Deploying services and retrieving content ................................................... 45
   2.2 Projects and activities ......................................................................................... 47
      2.2.1 The CASCADAS project ............................................................................. 47
      2.2.2 The ADDOaction project ............................................................................ 48
      2.2.3 Challenges and competitions ...................................................................... 50
         2.2.3.1 The semantic web service challenge ..................................................... 50
         2.2.3.2 The IEEE web service challenge .......................................................... 51
         2.2.3.3 The semantic service selection (S3) contest ......................................... 52
         2.2.3.4 The IEEE service cup .......................................................................... 52
   2.3 Summary .............................................................................................................. 53

3 Deployment Decision Making..................................................................................... 55
   3.1 Definitions and delimitation ................................................................................. 55
   3.2 Representation of information ............................................................................ 58
3.2.1 The characteristics of information in DDM .......................... 58
3.2.2 Necessary and available information .................................. 59
3.2.3 An information representing concept of parameters .............. 62
3.3 The DDM algorithm .................................................................. 66
3.4 Evaluation of the usefulness calculation .................................... 72
  3.4.1 Experiment 1 – selecting maximum usefulness .................. 72
  3.4.2 Experiment 2 – saving resources ..................................... 76
  3.4.3 Experiment 3 – extended music download scenario ............ 78
3.5 Summary .................................................................................... 79

4 Algorithmic complexity analysis ...................................................... 81
  4.1 Fundamentals and notation ...................................................... 81
  4.2 Algorithmic complexity of DDM .............................................. 85
  4.3 Influencing the algorithmic complexity ..................................... 88
    4.3.1 Effects of parameter selection for DDM ........................... 88
    4.3.2 Parameter selection methods .......................................... 89
      4.3.2.1 Selecting all decision parameters ............................ 89
      4.3.2.2 Selecting maximum x of all decision parameters .......... 90
      4.3.2.3 Selecting all common decision parameters ............. 91
      4.3.2.4 Selecting maximum x of all common decision parameters .......... 91
    4.3.3 Experimental evaluation of decision parameter selection ................................................................. 92
      4.3.3.1 Experiments without explicit upper bound ............. 93
      4.3.3.2 Experiments with absolute upper bound ............... 95
      4.3.3.3 Findings from the experiments ................................ 97
  4.4 Improving the average runtime performance of DDM .......... 98
    4.4.1 Considering maximum attainable usefulness .............. 99
    4.4.2 An example of DDM applying MAU ............................ 101
    4.4.3 Evaluation of runtime savings when using MAU ....... 105
  4.5 Summary .................................................................................. 111

5 Adaptability ..................................................................................... 113
  5.1 User-adaptability ................................................................. 113
    5.1.1 Selecting parameters ...................................................... 114
    5.1.2 Applying individual interpreters .................................... 115
    5.1.3 Adjusting parameter weights ......................................... 118
  5.2 Self-adaptability ................................................................. 120
    5.2.1 Provider reputation ...................................................... 121
### Table of Contents

5.2.2 Evaluation of provider reputation .................................. 123
5.2.3 Success rate ................................................................... 126
5.2.4 Evaluation of success rate .............................................. 127
5.3 Summary ............................................................................. 130

6 Conclusion and outlook .......................................................... 131
6.1 Summary ............................................................................. 131
6.2 Discussion ........................................................................... 132
6.3 Further work ....................................................................... 135

Bibliography ............................................................................ 137

Index ..................................................................................... 151

Eidesstattliche Erklärung ............................................................. 157
Abbreviations and Notation

The list below gives an overview of abbreviations and symbols used throughout this thesis. Each notation is accompanied with a brief explanation and the page number of its first occurrence.

#a: number of alternatives ................................................................. 85
#ap: number of adaptable parameters ..................................................... 85
#cp: number of common parameters ....................................................... 85
#dp: number of decision parameters ...................................................... 85
#p: number of parameters ..................................................................... 69
#ps: number of selected decision parameters ......................................... 85
#spa: number of service parameters of alternative a ............................... 85
a: alternative ....................................................................................... 64
A: algorithm .......................................................................................... 82
ACE: autonomic communication element ................................................. 47
ADDO: automatische Dienstvermittlung in dienstorientierten Architekturen ................................................................. 46
AJAX: asynchronous javascript and XML ................................................. 40
a_{max}: alternative with maximum usefulness ....................................... 102
ap: adaptable parameter ........................................................................ 63
AP: set of all adaptable parameters ......................................................... 69
API: application programming interface .................................................. 40
BPEL4WS: business process execution language for web services .......... 49
c_{1a}: costs of step 1 of the DDM algorithm selecting a subset of all decision and adaptable parameters ............................................. 86
c_{1c}: costs of step 1 of the DDM algorithm selecting a subset of all adaptable parameters and all common decision parameters .......... 86
An algorithmic approach to service and content deployment decision making

$c_{Sc}$: estimated costs of step 5 of the DDM algorithm .............................................. 87
CASCADAS: Component-ware for autonomic situation-aware communications, and dynamically adaptable services.............................................. 47
$c_{DDM}$: overall costs of the DDM algorithm................................................................. 87
$c_{DDM\_All}$: costs of the DDM algorithm selecting all matching decision parameters ................................................................................................................. 90
$c_{DDM\_AllCommon}$: costs of the DDM algorithm selecting all common decision parameters................................................................................................................. 91
$c_{DDM\_MAU}$: costs of the DDM algorithm applying when MAU ............................ 106
$c_{DDM\_MaxXOfAll}$: costs of the DDM algorithm selecting maximum $x$ of all matching decision parameters ................................................................................ 90
$c_{DDM\_MaxXOfCommon}$: costs of the DDM algorithm selecting maximum $x$ of all common decision parameters ................................................................................ 91
cf.: confer.................................................................................................................. 58
$c_{i}$: costs of step $i$ of the DDM algorithm ............................................................... 86
$const$: constant ...................................................................................................... 86
$cp$: common parameter .......................................................................................... 64
$CP$: set of all common parameters ........................................................................... 64
CPU: central processing unit .................................................................................... 26
DAML-S: DARPA agent markup language for services ........................................ 45
DARPA: defense advanced research projects agency ........................................... 45
DDM: deployment decision making ........................................................................ 27
d$p$: decision parameter ......................................................................................... 62
DP: set of all decision parameters .......................................................................... 64, 69
$E$: set of fields to store experiences ....................................................................... 63
et al.: and others..................................................................................................... 39
$f(x_i)$: function of the variables $x_i$ ....................................................................... 87
g(x): function of the variables x_i
---
HD: high definition
http: hypertext transfer protocol
IEEE: institute of electrical and electronics engineering
IP: internet protocol
iv: interpreted value
iv_p: interpreted value of parameter p
kBits/s: kilo Bits per second
MAU: maximum attainable usefulness
max: maximum
mv: measured value
n: name
N: natural numbers
n(s): number of DDM processes won by provider p
n_p: number of DDM processes provider p won
O(g(n)): O-noation for the function g(n)
OASIS: organization for the advancement of structured information standards
OWL: web service ontology language
OWL-S: web ontology language for services
p: decision or adaptable parameter
PAA: personal assistant agent
PCM: personal content manager
PDE: personal distributed environment
q_n: probability distribution
q_n(x): family of probability distributions for input data x of length n
An algorithmic approach to service and content deployment decision making

QoS: quality of service .............................................................. 41
RDF: resource description framework ......................................... 44
\( r_i \): learning rate ................................................................. 121
\( r_p \): reputation of provider p .................................................. 121
\( r_p(s) \): reputation of provider p after evaluating s DDM processes won by p .................................................. 122
S3: semantic service selection ....................................................... 52
\( s_{av} = 100 - sn \), percentage of value interpretation steps saved ............ 106
SA-WSDL: semantic annotations for the web service description language .................................................. 52
SD: service discovery ................................................................. 42
\( \sin \): sine .................................................................................. 73
SLA: service level agreement ....................................................... 41
\( sn \): percentage of value interpretation steps needed .................... 106
\( S_N \): usefulness scale .............................................................. 68
SOA: service oriented architecture ................................................. 39
SOAP: simple object access protocol ............................................. 51
\( sp \): service parameter .............................................................. 62
\( SP_a \): set of all service parameters of all alternatives ...................... 64
\( sr_p \): success rate of provider p ............................................... 126
SWS: semantic web service .......................................................... 50
\( t \): tolerance .............................................................................. 121
\( t_{DDM}^{*}(n) \): runtime of the DDM algorithm when the input data is of length n .................................................. 101
\( t_{DDM}(x) \): runtime of the DDM algorithm with input data x .............. 101
\( u(s) \): usefulness of an alternative after s parameters were evaluated ...... 99
\( u_a \): usefulness of alternative a .................................................. 69
UDDI: universal description, discovery, and integration................................. 42

\( u_m \): usefulness calculated based on measured service parameter values.................................................................................................................. 121

\( u_{MAU}(s) \): maximum attainable usefulness an alternative can achieve after \( s \) parameters were evaluated ........................................................................................................ 99

\( u_{max} \): usefulness of alternative with maximum usefulness .................. 102

URI: uniform resource identifier ................................................................... 40

URL: uniform resource locator........................................................................ 44

\( u_s \): usefulness calculated based on stated service parameter values........ 121

\( v \): value........................................................................................................ 62

\( v_p \): value of parameter \( p \) .............................................................................. 68

\( V_p \): set of values of parameter \( p \) ................................................................ 127

\( w \): weight .................................................................................................... 63

W3C: Worl Wide Web Consortium................................................................. 40

\( w_p \): weight of parameter \( p \) ........................................................................... 69

WS: web service ............................................................................................... 40

WSDL: web service description language....................................................... 43

WSML: web service modeling language......................................................... 52

XML: extensible markup language................................................................. 43

\( \pi \): number pi............................................................................................... 73
Publications

Parts of the work presented in this thesis have been published before. These publications are listed below.


1 Introduction

In this chapter we elaborate the field of our scientific work and determine which problem our research answers. We present the basic ideas of the approach we developed in order to solve the stated problem, and clarify the scope of this approach. Based on that, we explain how the work done contributes to theory and practice of the concerned scientific fields. Finally, we give an outline of the remainder of this thesis.

1.1 Problem statement

To illustrate the problem our research work solves we present two scenarios. The first one shows the autonomic management of advertising spaces at Times Square in New York. The second one describes the view of a user trying to obtain multimedia content. Based on the presented scenarios, we formulate the scientific problems that arise and the demands that have to be met in order to solve these problems.

1.1.1 The context aware autonomic advertisement scenario

The scenario described in this section shows the autonomic, i.e., user independent interaction of software entities that provide and consume services. Such entities reside in service ecosystems as explained in [1]. Given that a service ecosystem is large enough, it includes multiple entities providing and requesting similar services, which results in a competitive situation. Consequently, an autonomic entity has to choose one out of many co-operation partners that suits this entity’s needs most.

The service ecosystem of the context aware autonomic advertisement scenario comprises entities that provide and request services for the management of the advertising space at Times Square in New York City. According to [2], around 500000 people go through Times Square day by day, and 10 million viewers receive live pictures of the area via television. This results in the estimation that an advertising space at Times Square receives 1.5 million impressions per day. Figure 1.1 illustrates the density of the according advertising displays.
An algorithmic approach to service and content deployment decision making

Figure 1.1: Advertising space at New York Times Square [2].

In this public place, a large number of displays are operated by various advertisement management companies. Those companies pay for purchasing and maintaining displays, for renting space at the walls of buildings, and for the electricity needed to illuminate the screens. In contrast, other companies propose adverts and pay if they are shown on a screen. The payment details depend on context of the display where the advert is presented. Daytime, weather, or information about people and traffic in the vicinity of the concerned screen influence the amount of money an advertiser offers to pay. For the acquisition of context, various entities are providing the values of sensors, installed in the area, as services. An ad broker entity gathers the contextual information and has to decide which advert should be shown on which display in order to maximise the display operators profit. A typical decision making process looks as follows.
The ad broker has to decide what to display on the “Astrovision Screen” operated by News Cop. and Panasonic. Four offers for adverts are available: Fox News, an advert for Panasonic “Toughbooks”, Dow Jones ticker, and an advert for the Broadway show “West Side Story”. The ad broker entity queries different context providers and gathers information about the current date and time, the number of people in the vicinity of the Astrovision Screen, the number and kind of visible Bluetooth and WiFi devices, and the weather as well as weather forecast for the evening. Because there are several hundreds of people around, it is 11 a.m., only very few Bluetooth and WiFi devices can be detected, and the weather forecast predicts rain, the ad broker calculates the highest usefulness for displaying the “West Side Story” advert. The broker instructs the Astrovision Screen entity to display this advert and charges the advertiser Magic Theatre according to the acquired context as agreed in the concerned terms of business.

In the described scenario, one deployment decision needs to be made for each advert shown on each screen. The alternatives that have to be considered are all adverts proposed to be displayed for each screen. The parameters taken into account for making a deployment decision are the involved context data as well as additional demands the paying advertisers associate with a request to show an advert. Considering the facts that an advert usually lasts less than one minute and that more than 300 receptacles are available at Times Square [2], over 300 deployment decisions need to be made each minute. An autonomic deployment decision making system can be considered as a necessary tool to manage such an advertisement business.

1.1.2 The music video scenario

The music video download scenario describes a human user, Jane, who is obtaining multimedia content. Jane does so several times a day in order to listen to music or watch music videos. Parts of the media she likes to listen to or watch are available on her laptop because she retrieved it before. Other parts can be accessed online using the internet or any currently available network. Jane does not necessarily copy all obtained media content to her laptop. If this content is too large, if she intends to rarely use it, or if the right to store it costs additional money, Jane prefers not to store the concerned content permanently.
When searching for a special song or video, Jane usually discovers multiple alternatives how to obtain the desired media. Each of these alternatives comes with individual conditions concerning, for example, the price, quality, version, or size. Depending on her current situation, Jane has various preferences based on which she selects one of the available alternatives. For instance, she prefers alternatives with a minimum price, with a music sampling rate of 192kBits/s, with audio and video content, and versions with long duration. The following example illustrates one typical process of obtaining media.

Jane is at home, sitting at her desk, and working with her laptop. She feels like listening and watching a video of the Beatles song “Norwegian Wood”. So, she searches video providing platforms in the internet. Starting at www.metacafe.com, Jane only can find cover versions. She decides to search again at www.jamendo.com and finds no video matching there. Her next choice is www.myvideo.de which also leads to no search results. After that, Jane queries www.youtube.com and finds seven videos that come into consideration. She evaluates all of them manually by looking at the video duration (Jane prefers long versions), the video quality (Jane avoids HD videos to save bandwidth and CPU), the user rating, and the video’s date of publication (Jane prefers newer videos because they usually have less noise). The whole process of selecting the video that fits best to what Jane likes, takes more than 5 minutes, includes manual evaluation of more than 15 alternatives to obtain the desired content, and requires a multitude of decisions. Figure 1.2 depicts the situation of Jane.

![Figure 1.2: Multiple alternatives to obtain “Norwegian Wood”](image)
The type of content a user is trying to obtain is not restricted to music videos. It can as well be lyrics, tablature, or notation of music, images, literature like books or papers, software like code libraries or applications, and so on. A supportive decision making system can help to ease the process of selecting one out of many alternatives to obtain content. It can choose the alternative that fits best to the user’s preferences, in short time, and with minimum need for user intervention. A scientific approach to such a system is presented in the course of this thesis. In the following, the scenario described in this section will be used in order to illustrate details and examples because it is broadly applicable to various kinds of content, easily understandable without extensive domain specific knowledge, and of comprehensible complexity so that it can be used to describe concise examples.

1.1.3 Problem and demands

Based on the scenarios described above, we determine large networked computing systems as application domain for the work presented in this thesis. In such a network, a variety of service and content providers as well as consumers exists. Let us assume that for a consumer, multiple alternatives exist to obtain a desired service or content. Each of these alternatives is provided under individual conditions which concern for example quality of service or content delivery, quality of the service or content itself, properties and features of the service or content, or price the consumer has to pay to get the service or content. Let us further assume that the number of alternatives to obtain a desired service or content increases if more entities, i.e., service or content providers and consumers, participate in the network. Entities can be both human users as well as computing systems.

Looking from the perspective of a service or content consumer, the problem arises that it needs to be decided which of the available alternatives to obtain a service or content fits best to the consumer’s prerequisites and needs and should therefore be selected for deployment. We call this issue the deployment decision making (DDM) problem. With regard to the described application domain, we consider a set of demands when solving this problem.
• Due to the facts that a very large number of alternatives can be available to obtain a desired service or content, and that likewise the number of aspects considered as relevant by a consumer, calculating a deployment decision can be an algorithmically complex task. To be able to calculate a multitude of deployment decisions in short time, as for example needed in the context aware autonomic advertisement scenario (see section 1.1.1), our approach to solving this problem must work efficiently in terms of algorithmic complexity.

• In order to enable our approach to be applied by human users as well as computer systems, it is required that deployment decisions can be made autonomically, i.e., without human intervention. To this end, we demand that a decision making process executes independently from user actions. We allow manual configuration and adaptation only before or after a decision is made. Nevertheless, multiple decision making processes in sequence should be possible in an autonomic fashion if no such configuration or adaptation is wanted (see section 5.1).

• One further aspect to be considered in a computer network domain is the fact that the information a decision making process depends on is unreliable. This unreliability can have different reasons. Information can be incomplete or missing. It can be wrong because of unforeseen effects. Or it can be wrong because it has been altered purposely. In any of the three cases, we cannot avoid that unreliable information affects a deployment decision making process. Nevertheless, we demand that a decision can be made based on the information that is available. Furthermore, we demand that our approach enables the application of facilities to address the problem of unreliable information.

• In the computer networks we envisage as application domain for deployment decision making, numerous entities can participate. These entities can enter or leave the network arbitrarily, and they can change their preferences and needs in terms of offers or requests of services and content. Due to this fact, we demand that our approach to deployment decision making provides mechanisms for adaptation to such a dynamicity of the environment and users.
• The research work we do aims at providing a solution to come to deployment decisions that are in line with what a user, i.e., an individual person prefers. For that reason, we demand that our approach can be configured and adapted to meet the different requirements individual users might have. This includes being able to regard personal aspects of concern as well as enabling user specific interpretation of these aspects.

• The scenarios described in sections 1.1.1 and 1.1.2 are settled in different application domains. Various further areas of operation are imaginable for solving similar decision making problems. To achieve a scientific contribution beyond the dedicated domain of obtaining services or content in computer networks, we demand our approach to be flexibly applicable. The research work presented in this thesis is based on conditions, facts and investigations in the mentioned domain. Nevertheless, a principle possibility to utilise, or further investigate that work in additional areas should be provided.

In the course of our research work to solve the deployment decision making problem we referred to the demands stated above. Regarding each of them contributed to the approach we developed and present in this thesis.

1.2 Methodology and scope

To perform the research described in this thesis we proceeded according to the following methodology. First of all, we analysed the domain in which we settled our work, i.e., complex networked service- and content-rich computing systems. Thereafter, we figured out a problem within that domain. Along with investigating the state of the art concerning relevant topics and technologies, we isolated a concrete problem that we wanted to solve. We call this the problem of deployment decision making. In a next step, we determined the entities that are involved in deployment decision making processes. Based on this, we analysed requirements that need to be fulfilled to come to deployment decisions. These requirements include especially the nature and extent of necessary as well as available information. After gaining detailed insight into the domain, the task, and the requirements of deployment decision making, we defined a terminology to be able to describe the details of our work in a comprehensive manner.
Subsequently, we designed an algorithm that is capable of solving the problem we determined and described. According to the functions and processes of this algorithm, we defined all necessary metrics. To be able to test and improve the algorithm, we designed and implemented a corresponding prototype. This software enabled deeper investigation of components, metrics, interfaces, complexity, as well as applicability of our approach to deployment decision making. Throughout our further research, we continuously enhanced this prototype and the underlying theoretical principles by incorporating the consequences of experimental results. This led to adding new features and concepts, improving interaction interfaces, as well as optimising the average runtime performance.

The experiments we performed are of both, analytical and numerical nature. Depending on the aspects we investigated, we either deduced our findings in a theoretical manner, or we concluded from the results of experiments we performed on according data sets. In the latter case, we utilised artificially created data for simulations as well as real world data obtained by interacting with existing service and content providers. For each experiment we carried out, we designed an according scenario and defined all parameter settings. After performing the experiment itself, the scenario, the parameter settings and the results were stored to ensure reproducibility. Where applicable, analytical findings are supported and confirmed by experimental results.

Solving the deployment decision making problem requires to touch several adjacent tasks. Some of these tasks are addressed and solved. Others are out of the scope of our research. In the latter case, assumptions and definitions, concerning what is expected, are given instead. For adjacent topics, interfaces were investigated and are presented which explain required inputs and resulting outputs accordingly.

The core of our research is a decision making process in complex computing networks. As described in the scenarios of sections 1.1.1 and 1.1.2, the concerned decisions are related to obtaining services or content. We see autonomic systems, service ecosystems, and ubiquitous computing domains as exemplary environments where deployment decision making can be applied. Though we analyse service oriented architectures and technologies, we design our approach as much as possible independent of any specific technology in order to enable a broad and general application of our research results. Moreover, service discovery that takes place before a deployment decision is made, and service deployment as well as content retrieval themselves which are the consequences of a deployment decision,
are beyond the scope of our work. We regard these steps as available prerequisite and following result instead. More detailed explanations of the scope and delimitations of our research are given in the according sections of this thesis.

1.3 Contribution

This thesis contributes to theory and practice in the fields of network communication technology, autonomic computing systems, and service ecosystems. The theory is extended by formal definitions of first, the deployment decision making problem, second, entities as well as components that are involved in this task, third, metrics that help to assess and calculate the usefulness of a deployment alternative, and forth, an algorithm to solve the given problem accordingly. Moreover, we contribute by providing an analysis of the algorithmic complexity of our approach. We deduce in detail how each part of the algorithm affects this complexity. Because of our investigations and findings, the theory of the concerned scientific fields is also extended by a deep insight, looking at service deployment and content retrieval processes from the perspective of a requesting user. At the same time, this helps to build an understanding of the defined deployment decision making problem. Finally, we extend the theory by providing a flexible and extendable approach to solving the given problem.

We contribute to the practice of the concerned scientific fields because we implemented a running prototype of the approach we developed theoretically. In addition to this prototype, we designed several scenarios in different application domains. In line with the scenarios and the aspects we researched, we also created appropriate sets of example parameters and interpreters. A detailed explanation of the nature, properties, and effects of those parameters and interpreters is given in section 3.2. We also contribute by providing facilities to control the deployment decision making algorithm, where these facilities and their influence on decisions are evaluated extensively. Another contribution is the insight we gained into applicability aspects of our approach. Based on experiments utilising real world data and existing content providers, we collected experiences that reach beyond simulations with artificially created test data. Altogether, we carried out extensive experiments concerning various aspects of our deployment
decision making approach. The results of these experiments support, confirm, and illustrate our findings.

One major progress beyond the state of the art in the concerned scientific area is the fact that we provide a property for a user to select services and content automatically and meaningfully. In this context, the selection takes place regarding arbitrary aspects of concern which can individually be defined by each user. Furthermore, we enabled that the interpretation of these aspects happens in an individual manner. Both, aspects of concern as well as their interpretation can be exchanged and reassembled easily, depending on the needs and preferences of the user of our deployment decision making approach. Based on our research and development of a straight forward solution for the deployment decision making problem, we further progress beyond the state of the art by improving the efficiency of our approach. This is done by reducing the average runtime needed to compute a deployment decision, without a loss in quality of the calculated solution. More progress is achieved by the fact that our approach is, to a large extent, independent of underlying technology like web services, ontologies, or dedicated service discovery mechanisms. Instead, we designed our approach in a way that enables including it into arbitrary systems under the prerequisite that necessary data is available in accordance to all defined interfaces. Further details concerning prerequisites and interfaces for input and output data are given in the course of this thesis.

The deployment decision making approach we developed can be applied by human users as illustrated in the music video scenario (see section 1.1.2), and it can be used as a sub system of another computing system as shown in the context aware autonomic advertisement scenario (see section 1.1.1). The fields where a system based on our approach can potentially be applied comprise network communication technology, autonomic computing systems, service ecosystems and others. What we found out in the course of our work answers several questions. Nevertheless, many more questions arose. The results we achieved so far path the way for further research towards autonomic and self-managing systems, service based systems and architectures, as well as decision making processes in various domains.
1.4 Summary and structure of this thesis

In section 1 of the thesis at hand we introduced our research work by presenting two scenarios, firstly, dealing with context aware autonomic advertisement, and secondly, being concerned with the retrieval of music videos. Based on these scenarios, we determined the deployment decision making problem, saying that it needs to be decided in an efficient and meaningful manner, which of the available alternatives to obtain a service or content fits best to the consumers prerequisites and needs and should therefore be selected for deployment. In the following, we listed a set of demands we consider relevant when solving the DDM problem. We explained the methodology applied when investigating DDM scientifically, and we clarified the scope of our research work. Subsequently, an elaboration of our contribution to theory and practice of network communication technology, autonomic computing systems, and service ecosystems pointed out how our work extends the state of the art. Progress is achieved by providing a facility to solve the DDM problem, by defining all concerned terms, elements, as well as processes, by providing a prototype implementation of our DDM approach, and by extensively evaluating this approach experimentally.

The remainder of this thesis is structured as follows. In section 2, we summarise related work and briefly explain fundamental technologies. We exemplarily present two research projects and compile a list of current competitions concerning autonomic and service oriented computing. An approach to solving the DDM problem is presented in section 3. After clarifying the terminology used, we introduce a parameter concept to represent information. Subsequently, we explain in detail an algorithm we developed to compute deployment decisions. We illustrate with the help of simulations how this algorithm calculates usefulness values for deployment alternatives. In section 4, we present the results of an analysis of the algorithmic complexity of the DDM algorithm. We deduce the factors that influence, on the one hand, this complexity, and on the other hand, the quality of the deployment decisions made. By presenting and evaluating multiple methods for parameter selection, we equip the user with a facility to achieve a trade-off between spending computational effort and obtaining a deployment decision that reflects the user’s preferences. Thereafter, we introduce the principle of maximum attainable usefulness. Based on simulations, we demonstrate that this principle can save runtime while DDM still finds the deployment alternative with maximum usefulness. In
section 5, we present multiple facilities to adapt the behaviour of DDM. We distinguish between user- and self-adaptation and evaluate the effects of each facility experimentally. The experiments carried out are partially based on artificially created simulations and partially based on real world data we obtained by interacting with video providers in the internet. Finally, we conclude our work in section 6 by summarising the content of this thesis, discussing our findings and achievements, and presenting an outlook on issues that are not answered yet.
2 Related work and fundamentals

After we determined the deployment decision making problem in section 1.1.3, and exemplarily describing application scenarios where this problem exists, we will compile in section 2 an overview over research topics, projects, and further activities that are related to DDM. By creating this summary, we pursue the goal of clarifying the position of our research work done with regard to its origins, developments and the current state in computer science.

2.1 Related research topics

In this section we derive the position of DDM within related research topics by briefly explaining fundamental technologies and approaches.

2.1.1 Starting with artificial intelligence

In 1956, McCarthy, Minsky, Shannon, and Rochester organised the “Dartmouth Summer Research Conference on Artificial Intelligence”. By bringing together a community of researchers interested in fields like automata theory, learning mechanisms, and understanding of intelligence in general, this event became known as the starting point for artificial intelligence research [3]. The Dartmouth conference aimed at making computers calculate and simulate human behaviour.

Within the following years, remarkably optimistic results were achieved pursuing this goal. Accordingly, a dedicated field of sciences, addressing questions of artificial intelligence, was established. Despite ongoing progress, clear limitations of developed approaches and theories turned out. Successful solutions and applications were often limited to very restricted domains, i.e., toy worlds. The magnitude of information that had to be processed for solving problems was too large to be computed with available resources. The assumption arose that human behaviour involves unformalisable components [4]. Notwithstanding any scepticism, large communities or researchers assembled, tackling various aspects, problems, and application domains. Some of them pursued the original goals of achieving to simulate human behaviour. Others identified new goals in
creating supportive systems which should ease, augment, and make more comfortable interactions between humans, computers, and environments.

The DDM approach presented in this thesis is such a supportive system helping users to determine one out of many alternatives to deploy a service or to retrieve content in line with individual preferences and needs.

2.1.2 Ubiquitous computers pervading daily lives

To enable computer support for human environment interaction, devices were embedded into various situations [5]. At the same time, computing devices evolved towards being lightweight, small, and in many cases designed for dedicated purposes. In consequence, the number of computing devices, embedded in the environment as well as directly owned by human users grew rapidly [6]. In 1991, Weiser [7] shaped the term ubiquitous computing to describe that “specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence”. Out came a research direction which represents, amongst others, the idea that computing systems and devices are intertwined with people’s environments and every day’s lives, in a way that they are accepted as self-evident part of it. No special attention is necessary to utilise the services of those computing devices.

Systems performing DDM without explicit user intervention can leverage decision making processes in various situations. They might be seamlessly embedded in environments like the New York Times Square to support decision making processes of different kinds (see section 1.1.1). A vast amount of any sort of devices will be everywhere, perhaps interacting with each other. Accordingly, the dynamicity of entities in ubiquitous computing environments implies frequent changes of preferences, conditions, and situations [8][9]. Locations pervaded by computing devices, of course, enable almost all-embracing access to services and content. However, complexity of those systems is a consequence that needs to be regarded as well.

2.1.3 Handling complexity with autonomic computing

The problem of rapidly increasing complexity of a growing number of computing and communication systems is the core issue of the idea of
To resolve this issue, systems should be equipped with capabilities that enable working out problems in self-contained, user-independent manner. As a consequence, administration and operation efforts would be minimised for human users. Within the autonomic computing initiative [11][12], several self-* abilities were defined. These comprise, amongst others, self-configuration, self-optimisation, self-healing, and self-protection. Self-configuration means user independent adjustment of system components in order to achieve overall system behaviour according to high-level goals. Self-optimising systems change their actions over time based on observations, measurements, or any input that occurs. These changes lead to an improvement of system performance and efficiency. Self-healing comprises automatic diagnosis and repair of problems or malfunctions. The term self-protection circumscribes detection, prevention, and defence against attacks. The variety of self-* features can, in general, be summarises under the term self-management [13]. To incorporate self-* features into computing systems, scientists and developers apply ideas and principles from interdisciplinary fields of research. Methods of artificial intelligence are utilised for planning and deciding [14]. Many attempts to create autonomic computing systems are inspired by nature and biological archetypes [15]-[17].

Creating completely autonomic systems that solve complex tasks is a challenging undertaking. As autonomic computing research evolved, the question of how to quantify autonomic properties in order to investigate, differentiate, and classify systems more accurately arose. In consequence, the following five levels of autonomy were defined by Ganek and Corbi [10].

- **Basic.** No autonomic behaviour is implemented. Systems classified as basic are reactive only.

- **Managed.** Systems that are managed are reviewed and improved over time. Tools are utilised for data analysis. However, improvement and adaptation is implemented manually.

- **Predictive.** Predictive systems monitor their behaviour and possibly their environment. Reasoning based on monitored data leads to recommendations the system proposes to human users.
• **Adaptive.** A system not only recommends, but also executes actions based on monitoring and reasoning. Resources are provisioned and tuned by the system itself to optimise its own performance.

• **Autonomic.** For the class of autonomic systems, human users solely have to specify needs that have to be fulfilled. The system completely manages all required resources on its own. No user intervention is necessary at all.

According to Huebscher and McCann [18], the classification scheme for autonomy of systems was adapted to four classes in the following manner.

• **Support.** This class comprises systems that optimise one particular aspect of their overall function in an autonomic way. Consequently, this specific optimisation leads to an improved overall system performance.

• **Core.** Self-management concentrates only on one particular aspect of the overall system. This aspect, however, is essential, i.e., is the core of the system.

• **Autonomous.** In systems classified as autonomous several aspects work in a self-managed manner. Self-adaptation takes place according to observations of the system’s environment. Nevertheless, monitoring and improvement of the system performance itself is not put into practice.

• **Autonomic.** A system is called autonomic if it fulfils higher-level goals defined by human users without the need for any intervention.

According to both classification schemes, the categories of autonomic systems correspond to each other. Besides defining levels for autonomic systems, proposals to mathematically measure the degree of autonomy were made. For this purpose, Holzer compares information contained in the system’s control data with overall information available in a system itself [19].

To enable implementing complex systems of any level or degree of autonomic behaviour, a variety of frameworks was put into practice [1][20][21]. These frameworks offer a component model as well as an infrastructure for creating systems of interoperating entities. Equipping components with autonomic capabilities, however, is up to the framework user. In this context, component development does not necessarily need to
start from scratch. The CASCADAS framework as an example (see section 2.2.1) enables integrating already available legacy code [1]. Such legacy code can for instance be the DDM approach presented in this thesis. It can be used to contribute to a distributed, complex system of multiple interoperating entities for managing autonomously advertising space on Times Square as illustrated in the scenario in section 1.1.1).

2.1.4 Multiple entities in service oriented architectures

The creation of systems that comprise multiple, possibly autonomic, interacting components is leveraged by utilising service oriented architectures (SOA). In an SOA, each component can provide and consume one or more services. Complex applications are built by combining different simple services to more sophisticated functionality. Thus, SOAs are referred to as a reusable, modularised, and extendable architectural paradigm.

Several definitions exist that describe what a service is. According to Richter [22], a “service is a well defined accomplishment, which can be used as an element of one or more larger processing flows”. Following Keen et al. [23], “services are defined by explicit, implementation-independent interfaces. Services are loosely bound and invoked through communication protocols that stress location transparency and interoperability. Services encapsulate reusable business function.” Masak defines SOA as “a model of a system that is completely created out of autonomous services, the interaction of which takes place based on one and the same protocol, and the model always comprises the three roles provider, consumer, and broker” [24]. According to Hansen [25], an SOA “is a style of software architecture that is based on the use of services to create applications. [...] it implies a style of systems development whereby applications are composed by linking together individual services in a loosely coupled manner.” A multitude of open as well as proprietary technologies and frameworks exist that leverage SOA based software development [26]. A developer, however, does not necessarily rely on such dedicated tools. Utilising any programming language suffices to create an SOA as long as the resulting software systems comply with the above mentioned definitions.

Computing service deployment or content retrieval decisions, as an example, can be one dedicated service embedded in large scale SOAs.
Applying the DDM approach we present in this thesis can leverage or even automate to a certain extent the construction of functionality by selecting services according to a set of preferences and requirements.

2.1.5 Web services as exemplary technology

Despite a technological independence, web services (WS) are widely used to implement SOAs. Those web services are resources in the internet that provide certain functionality [27]. According to the World Wide Web Consortium (W3C), “a web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format [...]. Other systems interact with the web service in a manner prescribed by its description [...]” [28]. Each WS is globally identifiable by a uniform resource identifier (URI) [29]. This fact makes it accessible independent of location and time [30]. Common WS based architectures comprise consumers (requesting services), providers (offering services), as well as a repository (containing information about consumers and providers). One WS can take both, the role of a provider as well as consumer at the same time. This is the case when it offers a functionality that is composed of further functionality which is accessed as another WS [27].

One example illustrating the usage of web services is the Google AJAX Search API [31][32]. It enables access to search services by utilising an interface a developer can incorporate into a web page or application. A further example is the variety of web services offered by Amazon [33]. These include for instance

• fulfilment of online orders in a merchant’s online shop by automatically starting a service chain (pickup of the ordered good at the merchant location, initiating the shipping process, and billing the customer),

• acquisition and configuration of computational resources provided by Amazon,

• and a “Simple Queue Service” storing messages that are exchanged between two computers.

In accordance with the work we present in the thesis at hand, more but not yet implemented examples can be
• a web service offering deployment decision making within an SOA that manages advertisement on Times Square (see section 1.1.1),

• a content management system querying web services of content provider portals (see section 1.1.2),

• and a web service for sharing experiences about content retrieval processes within an online user community.

2.1.6 Ensuring quality when invoking services

To ensure appropriate availability of computational resources when transferring content or executing services within an SOA, quality of service (QoS) management mechanisms can be applied [34]. In this context, QoS describes non-functional capabilities of services, i.e., the conditions and properties of the environment services reside in. According to a more detailed definition given by Papazoglou [35], “QoS refers to the ability of the [...] service to respond to expected invocations and to perform them at the level commensurate with the mutual expectations of both its provider and its customers.” Various factors can represent the quality of a service which can be categorised into three groups [35]:

• **Performance and capacity.** This group covers factors that are concerned with amounts of computational resources like response time, throughput rates, or storage volumes.

• **Availability.** Factors that are related to recovery mechanisms, accepted downtime, or redundancy strategies are summarised in this group.

• **Security and privacy.** Factors like authentication and authorisation mechanisms, privacy policies, or encryption method belong to this third group.

Quality of service according to the above mentioned factors is ensured by negotiating service level agreements (SLA). These are comparable to a contract between parties, i.e., service consumers and providers, which defines details of accepted QoS factor values. Additionally, SLAs regulate penalties that are imposed to a party that violates an agreement. A service level agreement can either be of static or of dynamic nature. Static SLAs are valid for a defined period of time, and thus, ensure reliable conditions for
that period. Parties can plan their actions and resources accordingly. Dynamic SLAs are negotiated for each service invocation. Consequently, they are suitable for reacting to spontaneously occurring demands. Whether an SLA is static or dynamic, it is always created to ensure equal quality throughout at least one complete period of service usage.

DDM regards parameters, i.e., quality of service factors only during the process of service selection, and during the process of gathering experiences after service deployment or content retrieval. Nevertheless, accepting SLAs can be incorporated into a deployment decision making process in terms of parameters (see section 3.2.3 for details of parameter definitions). Accordingly, the presented approach can take into account offered SLAs when calculating, which of the available alternatives to deploy a service or retrieve content fits best to user preferences and needs. Besides SLAs, any aspect a user considers being relevant, and that can be represented as a parameter as defined in section 3.2.3, can be incorporated into DDM. This facilitates meaningful, user specific deployment decisions.

2.1.7 Discovering services and content

To be able to access content or services of any kind, with or without applying SLAs, they first have to be found. Location and conditions to access a service need to be known to the consumer. The process of figuring out available services in an environment is called service discovery (SD). A broad variety of approaches and protocols to put into practice SD exist. Each available method has special properties, depending on the kind of addressed service and the environment in which this service resides [36]-[38]. A widely used approach concerning web services is the universal description, discovery, and integration protocol (UDDI) [39] which is a standard recommended by the organization for the advancement of structured information standards OAISIS [40]. Within UDDI a service registry, dedicated to a defined network offers information about available services. This information comprises location and accessing requirements as well as descriptive metadata of available services. Moreover, mechanisms for managing a registry and for distributing web service information to other registries are defined. UDDI enables three kinds of data provision [41]:

- **White pages.** Information is sorted based on names of service providers.
• **Yellow pages.** Services are classified and information is sorted according to service classes.

• **Green pages.** Sorting criteria is the business case of service providers.

Besides approaches that enable utilising one dedicated SD mechanism, frameworks exist that attempt to combine multiple SD protocols. Doing so enables more flexible SD in heterogeneous networks that comprise various kinds of devices and services provided based on different technologies [42]. Additionally, contextual and semantic information can be incorporated to increase SD efficiency and to optimise resource consumption [43][44].

The process of discovering services and content is not an integral part of DDM. Instead, we assume that it takes place before a decision making process starts. The results of applying an appropriate SD mechanism are an input to the deployment decision making approach we elaborate in the thesis at hand.

### 2.1.8 Describing services and content

A necessary prerequisite for service discovery is to describe alternatives to obtain services and content appropriately. An extensive and accurate description is a solid base for the assessment of alternatives and selection done by DDM. Various description formats and languages exist which regard specific technical requirements, properties of described resources, and domains they are designed for [45].

A prominent and widely used approach for describing web services, for instance, is the web service description language (WSDL), which is a standard recommended by the W3C [46][47]. WDSL is based on the extensible markup language (XML) [48], and hierarchically represents information in tree-like structures. Web services are described by their providers according to a number of defined fields. Following Finger and Zeppenfeld [41], these fields include amongst others:

• **Message.** A message field is a description of exchanged data and comprises information about method names, as well as input and output parameters of the invoked service.
• **Types.** This field represents definitions of complex data types that are utilised when exchanging messages with the described web service.

• **PortType.** The information contained here associates service methods to ports.

• **Binding.** A binding defines the protocol used for communication between provider and consumer of the described service. That protocol can for example be the *hypertext transfer protocol* (http) [49].

• **Service and Port.** These fields describe the *uniform resource locator* (URL) [29] of a service, i.e., the location where the service can be found. Moreover, they include information about the port to which the provider binds the service.

A further approach to describing resources in the internet which are more general than web services is the *resource description framework* (RDF). It is a standard recommended by the W3C [50]. In RDF, information is represented as directed graph and is therefore suitable for expressing relations between resources. Resources are any objects which can be identified by a URI. To ensure unique identification of elements within a graph, each resource as well as each relation is described using a URI. Any RDF graph can be expressed as a set of triples, where a triple comprises a subject (URI of the source node), a predicate (URI of the relation between source and target node), and an object (URI of the target node). Triples, in turn, can be noted using XML. Hence, RDF is machine readable and can be further processed automatically [51]. According to the distributed nature of resources in the internet, RDF graphs describing different resources can be composed to a larger graph [52].

One more attempt to formalisation and structuring of machine readable, especially semantic information are *ontologies* [51]. Despite controversial discussions of an exact meaning or definition, in terms of computer science an ontology is commonly understood as specification of a conceptual system [53]. Such a specification describes concepts and relationships that can exist within a limited domain. Two or more ontologies can be merged to a larger one, if their syntax as well as semantic meaning match. The W3C recommends the *web ontology language* (OWL) as standard for representing an ontology [54]. Ontologies expressed in OWL can be mapped to RDF graphs and vice versa [55]. Based on OWL and the
2 Related work and fundamentals

DARPA Agent Markup Language for Services (DAML-S) [56], the web ontology language for services (OWL-S) is not yet recommended, but submitted for standardisation to the W3C [57]. OWL-S especially aims at service discovery and invocation by defining the following main parts:

• **Service profile.** This part describes what a service does.

• **Service model.** The model explains how a service works.

• **Service grounding.** Information about how to use a service is given in this part.

Due to this structure, ontologies expressed with OWL-S are well suited for representing service properties and QoS information for web services [58].

Besides WSDL, RDF, and ontologies, many more methods for information representation and service description exist. DDM is not restricted to use one of these mechanisms. Instead, any format to describe services or content can be used, as long as it is capable of expressing the properties of service deployment and content retrieval alternatives, as well as the conditions under which an alternative can be obtained, in terms of parameters as defined in section 3.2.3.

### 2.1.9 Deploying services and retrieving content

The term software or service deployment summarises activities related to the release, installation, activation, deactivation, update, and removal of software or service components as well as complete systems [59]. Content retrieval comprises obtaining content data from a source device in order to store it on a target device for usage (for example video download), or to use the content remotely (for instance video streaming). For service deployment and content retrieval, several approaches exist. All of them are designed according to special requirements, and dedicated to well defined environments [59]-[61]. In the following, we exemplarily describe, firstly, one content management approach, and secondly, one approach for service deployment.

The personal distributed environment (PDE) is an architecture enabling user-centric communication in heterogeneous networks [62][63]. It is the results of work within the scope of the mobile virtual centre of excellence [64]. According to Irvine [65], the PDE is a combination of
services and devices that a user controls. Two major entities are responsible for content management, namely the personal assistant agent (PAA) and the personal content manager (PCM) [66]. The former, PAA, seamlessly relays user interactions with one device to another device, if necessary. That means, despite possibly occurring changes in the underlying network structure, a user does not perceive a dissimilar situation. Moreover, the PAA decides which content to store on which device within PDE. The latter, PCM, stores and retrieves content of a user on different devices. Its functions comprise collecting and providing information about available content and its location, structuring and delivering content on demand, and presenting content related information to the user. Even though new content can be added to a PDE, the main task of the PCM is to manage a set of already internally available content where one and the same content is concerned several times. DDM is an approach for obtaining services as well as content. System internal rearrangement, however, is not foreseen as function. Instead, DDM processes can concern, but do not necessarily rely on the same content. Different deployment or retrieval decisions usually involve new service and content alternatives. While the PCM decides how to rearrange content to achieve optimal access and PDE system behaviour, DDM decides which alternative to obtain content and services fits best to current needs and prerequisites of a DDM user.

The ADDO framework is an example for automated service brokering in SOAs. One broker is responsible for discovery, selection, retrieval, and orchestration of services for multiple users [67]. In contrast to the ADDO approach, DDM is dedicated to the user side, i.e., no revelation of any data to a broker is necessary. Moreover, ADDO is restricted to using OWL-S descriptions and web service technology, which is no requirement for DDM. More details on the ADDO approach can be found in section 2.2.2, where we describe the related ADDOaction project.

The DDM approach we present in the thesis at hand does not address deployment or retrieval processes as such. Instead, we answer the question how to select one service deployment or content retrieval alternative out of many that fits best to user preferences and needs. By this means, DDM precedes service deployment or content retrieval. After a decision which alternative to obtain is made, the actual deployment or retrieval process follows. For our work, we assume that an appropriate mechanism to deploy a selected service or to retrieve selected content is available and will be executed as consequence of a DDM process. However, except for the
purpose of observing experiences (see section 5.2 for examples), DDM does not rely on the execution of deployment or retrieval processes at all.

2.2 Projects and activities

Depending on how far-ranging the word related is understood, several projects related to DDM exist. In the following, two exemplary projects are presented on a more fine grained level to foster an understanding of where to position our research on deployment decision making. Afterwards, an overview of frequently occurring challenges and competitions is given to illustrate the directions towards which service oriented and autonomic computing is evolving.

2.2.1 The CASCADAS project

The CASCADAS project is an example for an autonomic environment consisting of interacting entities which can provide as well as consume services and content [68]. Entities residing in a CASCADAS environment are independent, i.e., they can ignore each other, cooperate, complement each other, or compete with each other in order to fulfil individual tasks. Environments themselves can be distributed among multiple devices, and they can be connected to a network. In this respect, CASCADAS environments can be used as infrastructure on top of IP networks. System architecture and entity design enable several autonomic features (see section 2.1.3):

- Self-preservation and self-healing are facilitated by applying a supervision system that detects and corrects malfunctions or unfavourable situations [69]-[72].

- Self-adaptation is achieved by enabling context aware reactions to changes and incorporating gathered knowledge [73]-[76].

- Self-similarity is preserved because of an identical underlying component model for each entity [1][77].

Entities are circumscribed by the term Autonomic Communication Elements (ACE). Each ACE manages its own life cycle, which is independent from the life cycle of other entities. That means ACEs can be
created or destroyed on demand. Cloning as well as migrating ACEs from one environment to another is possible, too. Each ACE consists of two sections, namely a common and a specific part. The common part is identical for each and every component in an ACE based system. The specific part is always individual according to the functionality an ACE is designed for. This functionality may be arbitrary code that can be implemented using the Java programming language. At the time of writing this thesis, a toolkit for ACE development based on Java is available [78]. Java Micro Edition and Android are supported as well for integration of mobile devices like smart phones.

A system based on the ACE approach, as it was developed in the CASCADAS project, might for example put into practice a context aware autonomic advertisement scenario similar to what we describe in section 1.1.1. In such a scenario, a number of ACEs offer adverts to be displayed and paid according to individual contextual situations. Another set of ACEs represent screens, i.e., advertising space available for rent. Context provider ACEs expose information about environmental conditions and observable data. One ACE corresponds to the ad broker performing DDM in order to find out which advert to display on which screen. However, the CASCADAS toolkit provided for ACE based software development does not offer any components that are able to solve the deployment decision making problem we state in section 1.1.3. It rather equips developers with a basic framework being capable of incorporating legacy code [1]. To this end, an implementation of the DDM approach we present in this thesis might be utilised within an ACE based system if the need for such functionality exists. Following that, ACE based systems are one example of an autonomic and service oriented environment, where the results of our research can be applied.

### 2.2.2 The ADDOaction project

ADDOaction and its predecessor ADDO are concerned with automated service brokering in service oriented architectures [79][80]. These two projects are presented as examples for the entire process of web service deployment with respect to service level quality constraints. Goals of ADDO and ADDOaction are the development of an automatic service discovery algorithm which takes care of QoS aspects, as well as a framework that enables automatic integration and management of services
within a SOA based application. An ADDO framework is available for application and development as Java implementation.

The two projects attempt to achieve self-healing and self-adaptation properties by reacting user independently to missing services, faulting services, and changing demands in Quality of Service. The core component of the ADDO framework is a service broker which is responsible for registering and deregistering services to a service register [67]. Four sub components constitute the service broker. Namely these are description manager, matcher, stub creator, and stub pusher.

- The description manager distributes service descriptions to a broker internal service register and knowledge base.

- The matcher performs service discovery if a service request or service offer occurs. It also takes care of matching requested and offered QoS aspects.

- Stub creator and stub pusher then enable access to the matched service so that it can be invoked by the requester and executed by the provider.

ADDO and DDM address a related task in terms of service selection. A variety of similarities and differences, however, exist. While the ADDO framework handles the complete service deployment process, DDM is specialised solely on the selection of the best suited alternative to obtain a service or content. ADDO is bound to dedicated technologies (OWL-S for ontologies which represent service properties, services have to be web services, BPEL4WS [81] is used for service aggregation). DDM is not restricted to technologies, languages or services as such. Our approach can decide upon alternatives to obtain services as well as content. Moreover and with regard to interfaces that have to be met, DDM can be integrated into legacy software systems. ADDO is a System centric approach for multiple users. One global integration manager is responsible for service trading and negotiation within the whole service infrastructure [82]. In contrast, one individual DDM system for each user avoids the need for exposing private data to any central entity and facilitates incorporation of personal experiences as well as user centric configuration. In ADDO, on the one hand, different parameters are used for searching and matching purposes. Explicit interpretation of those parameters in terms of usefulness does not take place [82]. A ranking is created, which is based on matching information like the number of matching parameters and the number of
required plug-ins. Additionally, potential rewards and penalties of considered SLAs are taken into account [67]. On the other hand, DDM evaluates how useful each involved parameter value is, and compares all alternatives with regard to all calculated usefulness values. Considering the fact that number and kind of involved parameters are not restricted, the comparison of alternatives in DDM is a more differentiated approach than rankings created in ADDO.

All together, the research concerning ADDO and DDM are two pieces of work in a similar field of application. Nevertheless, both approaches differ in terms of goals they envisage, as well as in terms of tasks they address. On the one hand, ADDO aims at providing a complete solution for the entire process of service deployment. This solution is restricted, however, to the usage of a dedicated set of technologies (like web services and OWL-S). On the other hand, DDM specialises on determining a meaningful decision of which alternative to obtain a service or content fits best to individual user needs. This approach is independent of specific service oriented technology and can thus be adapted to specific implementations, or incorporated into existing systems when needed. For both approaches, working prototype implementations exist based on the Java programming language.

2.2.3 Challenges and competitions

In this section we list a variety of periodically occurring challenges that are concerned with the investigation, development, and usage of services in service oriented environments. These events visualise trends and up to date questions that are related to service oriented and autonomic computing.

2.2.3.1 The semantic web service challenge

The semantic web service (SWS) challenge is an annual event that pursues the goal of automating the discovery, mediation, and composition of web services [83]-[86]. Special emphasize is put on semantic annotations. The organisers of the SWS challenge provide problems that have to be solved by contestants during a number of workshops. Besides isolated research, this kind of organisation enables personal discussions on
similar applications. All exposed problems have in common that web services have to be chosen and invoked using a correct, task dependent message sequence. Participating developers have to implement all software as web services. Information exchange within the software system of the SWS challenge is defined to take place utilising SOAP messages [87]. Web service descriptions have to be ontologies denoted in OWL, while semantic annotations to web services are provided in form of WSDL descriptions.

Following these descriptions, it becomes clear that the SWS challenge is strictly limited to the usage of web service technology. In contrast to the goals of this contest, the DDM approach presented in this thesis is, to a certain extent, technology independent and can be combined with web services as well as software using according APIs of service and content providers. Not only service deployment, but also content retrieval decisions are considered by DDM. Moreover, service descriptions in DDM are not restricted to any information representation format as long as they can be expressed in terms of parameters and processed by according interpreters (see section 3.2.3).

In summary, the SWS challenge is not dedicated to find one solution to a specific problem at all. Rather the competition should find different individual solutions which lead to discussions among developers. This, in turn, should leverage research concerning service oriented architectures in general.

### 2.2.3.2 The IEEE web service challenge

Similar to the SWS challenge, the IEEE web service (WS) challenge is an annual competition [88][89]. Its focus lies on service discovery and composition of web service chains. Developers and researchers are asked to combine simple web services to higher-level functionality. The WS challenge is limited to utilising web service technology and OWL ontologies for representing taxonomies and concepts. The parameters, upon which a proposed solution is assessed, are invariably fixed to least response time, highest throughput, shortest web service chain composition length, and least amount of execution steps inside the service orchestration [89].

In contrast to the WS challenge, the DDM approach is not limited to assessing one defined set of parameters to judge the usefulness of an alternative, i.e., solution. Instead, a user can imply any number and kind of aspects he or she considers to be relevant (see section 3.2.3). Furthermore,
single services and service chains are not distinguished in DDM. Throughout all WS challenge solutions, service selection is settled on the server side where the contestant software has to find a service composition based on service requests. DDM performs decision making on the client side, i.e., directly at the service requester. This enables individually storing experiences, exchanging parameters and interpreters, and adapting the system configuration (see section 5). Moreover, DDM evaluates service and content offers based on user requests. While the WS challenge is most concerned with finding efficient service composition solutions and dedicates itself to web service technology, DDM focuses on finding meaningful service deployment and content retrieval decisions while being independent of specific technology.

2.2.3.3 The semantic service selection (S3) contest

Like the events described in sections 2.2.3.1 and 2.2.3.2, the S3 contest is an annual competition with the purpose of leveraging web service selection technologies with special focus on semantic information [90]. Its goal is to evaluate and improve the retrieval performance of publicly available semantic web service selection approaches. In this context, the S3 contest is limited to a set of dedicated technologies and information representation formats like OWL-S, web service modeling language WSML [91], and semantic annotations for the web service description language SA-WSDL [92]. For the evaluation of submitted solutions, metrics like recall and precision of service selections, as well as average query response time are assessed.

Here again, being not restricted to web services and enabling flexible usage of assessment parameters are features we take into account in DDM. Even though we consider the computational complexity of DDM, the overall query response time is not regarded in our approach. This measure depends to a large extent on service and content providers which we cannot influence at all.

2.2.3.4 The IEEE service cup

The IEEE service cup takes place annually and aims at determining solutions as well as problems with respect to service oriented architectures
applied in the real world [93]. The contestants have to submit a scientific paper and provide online access to a demonstration. In contrast to the challenges described in section 2.2.3.1 to 2.2.3.3, each service cup participant has to choose their own task for the required demonstration. The organisers of the contest only provide a general topic, which is cloud computing for the service cup 2010. The contestant has to show novelty in the demonstration that illustrates the solution of the task he or she formulated his- or herself.

Because of the possibility to choose application details and technological basis freely, the service cup is an opportunity to place approaches that do not fit into the concepts of any of the challenges mentioned in section 2.2.3.1 to 2.2.3.3. We, however, have not taken part for the time being, but contributed the results of our research to the scientific discourse by publishing articles and presenting at conferences and workshops. A list of publications related to our research on deployment decision making can be found in the beginning of this thesis.

2.3 Summary

In section 2 we compiled an overview of fundamentals and related work. Firstly, we shortly explained basic technologies of ubiquitous, autonomic and service oriented computing. We subsequently reconstructed the origin of each of these principles and deduced the emergence of the deployment decision making problem we address in this thesis (see section 1.1.3).

Secondly, we briefly presented two related research projects, CASCADAS in section 2.2.1 and ADDOaction in section 2.2.2. The CASCADAS project focused on the creation of a service infrastructure in order to enable autonomic communication and user independent operation within distributed complex computer networks. The results of CASCADAS are one possible technology for developing systems in which our approach to deployment decision making can be applied. The ADDOaction project illustrates the position of our research within a wider context. This context comprises the complete process of service management. DDM is specialised on performing one part of this process, the selection of one suitable alternative to obtain a service or content.

After illustrating the relation of DDM to other work, we compiled a list of challenges and competitions that are concerned with autonomic and
service oriented computing. These show current trends of the research domains we associate with our work. Even though we thoroughly investigated a broad field of the scientific discourse that took and takes place in the past up to now, we might have missed single approaches. However, we have referenced within this thesis each source of information we utilised for our research. Together with the fundamentals and related work presented in this section the reader is equipped with detailed indications to comprehend the DDM approach we present, and to position our research within computer science.
3 Deployment Decision Making

In this section we describe the basic ideas and principles of how we address the deployment decision making problem. Formal definitions of terms, elements, and processes are given. Our work is delimited against adjacent topics. After explaining our concept of how we represent information, we introduce the DDM algorithm. Based on this algorithm, we describe our way of calculating the usefulness of a deployment alternative. Implications of selecting an alternative based on calculated usefulness values are investigated experimentally.

3.1 Definitions and delimitation

The scenarios described in sections 1.1.1 and 1.1.2 illustrate the deployment decision making problem we address in this thesis. This problem can occur in computer networks like the Internet, the intranet of, for example, a company, or even in small ad hoc networks as they can be formed by spontaneously connecting two or more devices. With the term network we denote two or more computing devices that are connected in a way that they can communicate, i.e., exchange data. A connection between two devices does not need to be permanent or direct. It may be temporarily available and data may be exchanged indirectly utilising other devices in between.

Two of the purposes of computer networks as we consider them in our work are the utilisation of services and the exchange of content. In the context of DDM, a service is a piece of software that actively fulfils a certain task. Therefore, an input can be required and an output can be made. Services may be utilised by entities other than the service itself. Those entities can be human users or further services. Examples for services are executable computer programs, the provision of weather information via internet, or a web service that offers a facility to use a search engine for searching the internet (see section 2.1.5). A network in which services are provided and can be utilised is called a service ecosystem. Beside services, a network can enable access to content. With content we refer to data that does not fulfil a task actively, but can be used to fulfil a certain purpose. Content can be distributed over the network by transferring it from one device to another. Content can be sent, received or used by human users or
services. Content itself does not need any input to be used. Nevertheless, a service that provides content may require input anyway. Examples for content are pictures, e-books, or music videos as requested by Jane in the music video scenario (see section 1.1.2). Despite the formal separation given above, the boundaries between service and content are blurred. It is, for example, not exactly clear if stock prices provided on the homepage of a bank are content or if they should be considered as part of a stock price information service. However, for solving the deployment decision making problem given in section 1.1.3 by applying the approach we describe in this thesis, it does not matter if data is classified as service or content. Both kinds of data can be handled accordingly.

Beyond connected devices, a network comprises entities that are able to act and react. Those entities can be human users or computing systems. They interact by utilising the underlying network. If an entity offers either a service or content to be utilised by others, that entity is called a provider. If an entity requests a service or content from others, it is called a consumer. Within the scope of DDM, entities can be provider, consumer, provider and consumer, or neither provider nor consumer of services and content.

If a consumer wants to obtain a special service or content, we call the concerned data desired service or desired content. In order to find available services and content within a network, so called content or service discovery mechanisms are applied (see section 2.1.7). How service discovery works and which technologies are available is beyond the scope of our research. Several service discovery methods exist where each of them fulfils a set of requirements and is applicable under certain circumstances. For DDM, we assume that an appropriate service discovery mechanism exists which can be utilised before a deployment decision is made. We take the result of applying a service discovery mechanism as input for DDM instead. Once a desired service or content is found and has been selected, this service can be deployed or this content can be retrieved. Service deployment means copying or moving a service from a source to a target device, or utilising a service remotely. Remote usage of a service takes place when the concerned service is executed on the source device and only the output results are transferred to the target device. Deployment also includes configuring and installing the service appropriately [59]. Content retrieval means to copy or move content from a source to a target device, either before (for example video download) or during (for instance video streaming) the usage of the concerned content. Both processes, service deployment as well as content retrieval are beyond the scope of our
research. We instead concentrate on the decision which service or content should be obtained. In the context of DDM, a source device is the device where a service or content can be obtained from, and a target device is the device where a service is deployed or content is copied or moved to. A device in general, is a computing system that can be connected to other computing systems within a network. It is capable of storing services and content, and it can be capable of executing services or using content for purposes of the device’s user. A device can be of physical or of virtual nature. Physical devices can for example be servers, personal computers, or smart phones. Virtual devices can be virtual machines that emulate a physical device.

Within a network, multiple possibilities may exist to deploy a desired service or content. This can for example be due to the fact that various copies of exactly the same content are available on different devices, or that similar services exist which all fulfil the requirements desired by the requesting consumer. We call each of these possibilities a deployment or retrieval alternative. Such an alternative is offered and can be obtained under individual conditions which represent circumstances of the transfer of the concerned service or content, as well as properties of the service or content themselves. In order to inform a consumer about the conditions of an alternative, the provider needs to describe them accordingly. For this purpose, content and service description mechanisms are utilised (see section 2.1.8). Within the scope of our scientific work, we apply service description. However, we do not research this field. We rather utilise a notation that is suitable to represent the information that is part of our scenarios and experiments. Other notations in combination with DDM are conceivable. In our approach, conditions of alternatives are denoted as so called service parameters which are made available by service or content providers. Together with decision parameters, representing aspects of concern of a service or content consumer, and adaptable parameters, corresponding to experiences a DDM system collected during former decision making processes, service parameters are interpreted. Based on this interpretation, a usefulness value is calculated for each alternative, which should reflect how useful an alternative is for a user in the context of a dedicated decision making process. A detailed description of the various kinds of parameters can be found in section 3.2. Information on the usefulness calculation is given in section 3.3. The explanations presented in this section have the purpose to establish a general understanding of terms
we use throughout this thesis. Further definitions are given in according sections.

3.2 Representation of information

In this section we elaborate characteristics and requirements of information that is required for DDM. Based on that, we explain how we represent information within our approach utilising a parameter concept.

3.2.1 The characteristics of information in DDM

In order to come to meaningful service deployment or content retrieval decisions in the computing network domain described above, we need to collect, represent and evaluate related information. Therefore it is necessary to analyse the nature of the information DDM needs to process. If we consider a computer network environment (cf. section 1.1) comprising numerous entities that provide and consume services as well as content of different kinds, we can determine a set of characteristics that constitute the information we have to deal with.

- **Large amounts of Information**. The networks we consider for DDM are not restricted in size. That means, plenty of entities can participate by providing multiple services or content. If we regard the internet, providers and consumers can exist everywhere in the world. Of course we cannot consider all information concerning every reachable entity, because today’s computing systems are limited in computational resources. Even if we look upon the DDM problem on a smaller scale, large amounts of information have to be handled. For example in the music video scenario described in section 1.1.2, Jane queries only a small number of potential content providers but might get many alternatives among which she can choose. Following that, two requirements turn out to be important in order to handle the issue of large amounts of information. First, an approach that solves the DDM problem needs to be capable of handling a certain amount of information efficiently, and second, a facility to restrict the amount of information that is regarded within a decision making process is necessary.
• **Technical Information.** In the domains where DDM can be applied, computing systems participate in communication processes. That means, a human user may interact with a computing system or one computing system may interact with another one. Furthermore, it is required to exchange technical information about the transfer of services or content after a deployment or retrieval decision has been made. This technical information can for instance comprise details of network infrastructure, data transfer protocols, or data input and output interfaces for service utilisation. Additionally, DDM might be applied to services or content of technical nature themselves, like music video files as described in section 1.1.2. From these facts, we can deduce that in the context of DDM, we have to consider information that is, at least partially, of technical nature. That means on the one hand, it is not necessarily human readable and it might require deep technical knowledge to be understood semantically. On the other hand that means it might be machine readable to a certain extent.

• **Unreliable Information.** Considering numerous independent entities, each of them having individual interests, as well as a highly complex, distributed and heterogeneous network infrastructure, it is clear that information that is exchanged under these circumstances can be unreliable. Unreliability can have different reasons which are explained in more detail in section 5.2 where we describe self-adaptation abilities of our DDM approach. For the purpose of representing information we need to regard that unreliability is possible and might be reflected by personal experiences.

To be able to develop an approach that solves the DDM problem in a meaningful, efficient, and autonomic manner, we need to represent information in a way that considers the above mentioned characteristics that information, a decision making process relies on, can have.

### 3.2.2 Necessary and available information

In order to select one out of many alternatives to deploy a desired service or to retrieve a desired content, several data are required. First of all, a DDM system that should help with such a selection needs to know what service or content is desired. A name, description, or identifier needs to be
An algorithmic approach to service and content deployment decision making

provided by the DDM system user accordingly. Next, the system must get as an input a set of alternatives that match the name, description, or identifier of the desired service or content. The size of this set influences the range in which the DDM system has to select a meaningful service or content. Nevertheless, it also affects the computational resources necessary to come to a deployment decision so that the number of alternatives to be evaluated might be limited by the user. In order to select an alternative that fits best to a user’s needs and prerequisites, the DDM system requires information about what matters for the user within the context of the current decision making process. Consequently, the user has to specify all aspects that should be regarded. Due to the individuality of users, not only aspects of concern but also the way how these aspects matter for the user need to be clear. Only if the user expresses correctly his or her demands, decisions calculated by DDM are meaningful. To enable the DDM system to assess alternatives regarding defined aspects of concern, each alternative must expose information about how it fulfils those aspects. Combining the information about what matters to a user in which way, and how alternatives meet these preferences and requirements leads to a rating that judges how useful each assessed alternative is within the context of a decision making process. Finally the deployment decision itself needs to be made by selecting the alternative that is most useful, and information about the name, description, or identifier of this alternative has to be passed to the user that desired the concerned service or content.

We can summarise the information required to come to a deployment decision as follows. The user applying DDM needs to specify the desired service or content, the maximum number of alternatives to be assessed, aspects that matter for the user and the way how these aspects influence the usefulness of an alternative for the user. Providers need to enable access and identification of services and content they offer, descriptions of the properties of the offered services and content, as well as descriptions of the conditions under which a service or content can be obtained. The DDM system itself has to inform the user about which alternative has been selected as result of a DDM process.

After analysing what information is necessary for DDM, we continue with listing what information is available. From the perspective of a user, we assume that all required information is available because the user is capable of expressing what service or content is desired, how many alternatives should be regarded, and which aspects matter in which way. Passing the information about the selected alternative after a decision
making process from the DDM system to the user can be ensured as well, because we designed our approach to DDM accordingly. As described in section 3.1, we assume that alternatives to obtain a desired service or content can be identified by applying an appropriate service discovery mechanism. Following this assumption, information about identification and access of alternatives is available. Information describing the properties of alternatives to obtain services or content, as well as the conditions under which these alternatives can be obtained is not granted for all DDM processes. This information has to be denoted by service or content providers. Nevertheless, depending on the scenario where DDM is applied, various data that describe offered services or content are available and can be used to determine properties and conditions of alternatives. Music videos offered by video providers in the internet (cf. section 1.1.2), for example, are accompanied by a set of information like video duration, artist, song title, user ratings, file size, file format, and so on. This information can be utilised by a DDM system under the prerequisite that it matches to the aspects the DDM system user defined to be relevant. Due to the fact that these aspects can comprise any imaginable issue, a DDM system needs to handle the problem that not each defined aspect of concern has a corresponding description denoted by service or content providers for each assessed alternative. Furthermore, not every information given by a provider will be evaluated in each decision making process. The approach we present in this thesis is capable of handling this issue. Deployment decisions can be calculated as long as there is a match between defined aspects of concern and service or content descriptions. More matches can result in decisions that better represent user needs and requirements. In addition to all available information listed above, a DDM system can collect experiences during the time of its operation. These experiences can concern anything that is involved in decision making processes, i.e., anything that is based on any information a DDM system can or could access. In the music video scenario (see section 1.1.2), experiences can for instance be information about the success of video retrieval processes depending on the video provider.

Information available for DDM can be summarised as follows. Any information that is required from a user is available. This includes the specification of a desired service or content, the definition of aspects of concern and their interpretation, as well as a limitation of the number of alternatives to be assessed. Information about the selected alternative after DDM is given by the DDM system. Furthermore, a DDM system can
collect experiences about decision making processes it was involved in. We assume that alternatives can be found and identified. The description of alternatives is available in some, but not in all cases. Denoting properties of services and content as well as conditions of obtaining alternatives is an effort that has to be spent by providers in order to be able to apply DDM.

### 3.2.3 An information representing concept of parameters

Any information that is involved in a DDM process needs to be denoted. Properties and conditions of alternatives can be expressed using content and service description formats (see section 2). In the case that the user of a DDM system is human, aspects he or she considers important are specified in a human readable format. However, content and service descriptions as well as specifications of aspects of concern have to be processed by a DDM system which requires machine readability. To represent information in our approach to DDM, we developed a concept of different parameters that considers the duality in readability. We utilise the following types of parameters.

- **Service parameter** (*sp*). A service parameter represents information about the conditions under which an alternative can be obtained. A condition describes circumstances of the transfer of the concerned service or content, or properties of the service or content. Multiple service parameters can be used to describe different conditions or properties of one alternative. Service parameters are made available by service or content providers and are individual for each alternative. They consist of a name *n* and a value *v* (see Figure 3.1):

  $$ sp=(n, v). $$

The name identifies the *sp* and is used to find matching decision parameters. Value is a container that stores arbitrary information about the condition the concerned *sp* represents.

- **Decision parameter** (*dp*). A decision parameter represents an aspect that is regarded when calculating a service deployment or content retrieval decision. Decision parameters are the counterparts of service parameters and are specified by the user of our DDM approach. Each time an alternative is assessed, our DDM approach
tries to find a corresponding \textit{sp} for each \textit{dp}. If a match between \textit{sp} and \textit{dp} is found, a usefulness of the concerned \textit{dp} according to the regarded alternative is calculated. A \textit{dp} consists of the fields name \textit{n}, value \textit{v}, measured value \textit{mv}, interpreted value \textit{iv}, and weight \textit{w} (see Figure 3.1). Additionally, it comprises an interpreter:

$$dp=(n, v, mv, iv, w).$$ \hspace{1cm} (3.2)

The name identifies the \textit{dp} and is used to find matching service parameters. Value is a container to which the denoted value of a corresponding \textit{sp} is copied. Measured value is a container that stores the actual value of the condition the regarded \textit{sp} and \textit{dp} represent. This actual value might be measured during or after a deployment or retrieval process. It may differ from the value of the \textit{sp} that is denoted by a service or content provider. The interpreter of a \textit{dp} is a component that maps the value of the corresponding \textit{sp} to a well defined scale of usefulness which is defined in section 3.3. The result of this mapping is called interpreted value. The weight of a \textit{dp} is a factor that is utilised to increase or decrease the influence this \textit{dp} has on the overall usefulness of an alternative. Within our approach to DDM, decision parameters as well as their interpreters can be exchanged individually by users.

- \textbf{Adaptable parameter (ap).} An adaptable parameter is a decision parameter extended with a facility to store data for longer than one decision making process. The data it stores can be initialised and reset. An \textit{ap} is equipped with the ability to learn which means, to change the stored data according to experiences it collects during former decision making processes. In contrast to decision parameters, adaptable parameters are not compared to service parameters. They rather calculate their usefulness by interpreting the experiences stored. An \textit{ap} consists of the same fields as a \textit{dp} and includes an interpreter as well. Additionally, it comprises methods for initialisation, reset, and learning as well as fields \textit{E} for the experiences stored (see Figure 3.1):

$$ap=(n, v, mv, iv, w, E).$$ \hspace{1cm} (3.3)

Value here corresponds to an input that can be passed to the \textit{ap}. The fields \textit{E} contain all experiences stored. Measured value can
optionally be used to represent the experiences resulting from the most recent DDM process after adaptation. The learn method incorporates all experiences made during the current DDM process into the experience fields $E$. Currently made experiences, i.e., all denoted service parameter values and all measured values of the winning alternative as well as information about the deployment success, are passed to the learn method as a parameter. Within our approach to DDM, adaptable parameters as well as their interpreters can be exchanged individually by users.

- **Common parameter ($cp$)**. A common parameter is a $dp$ that fulfils the requirement of having a corresponding $sp$ for each alternative $a$ within the context of one DDM process.

$$cp=(n, v, mv, iv, w). \quad (3.4)$$

$$\forall a : CP = SP_a \cap DP. \quad (3.5)$$

$a$: Deployment alternative.  
$CP$: Set of all common parameters.  
$SP_a$: Set of all service parameters of all alternatives.  
$DP$: Set of all decision parameters.

Following this definition, common parameters are not an additional construct to represent information. Rather they are a subset of all decision parameters that are involved in making a deployment decision (see Figure 3.1). Because of the fact that service and content providers are free to choose which and how many service parameters they denote for a deployment or retrieval alternative, it cannot be granted that all alternatives that are assessed within a DDM process have the same number and kind of service parameters. This implies that alternatives might be judged upon a dissimilar base of assessment. By providing the construct of common parameters, we enable the user of our DDM approach to overcome this obstacle if he or she prefers equal conditions for assessing different alternatives. Further details on the usage of common parameters are given in section 4.3.2. All kinds of parameters occurring in DDM are visualised in Figure 3.1.
As discussed in section 2, various service and content description formats exist which can be used to denote service, decision, and adaptable parameters. However, we do not restrict ourselves to one special format because different notations are advantageous in the different domains where DDM can be applied. Our approach is designed in a way that allows any suitable format to be applied because the DDM algorithm is independent from the notation of represented information. We achieve this by using only interpreted values for calculating deployment decisions. These interpreted values are all limited to a well-defined range. To adapt to different description formats, the user of our DDM approach needs to adapt the format of all utilised parameters as well as their interpreters (see 5.1.1).

Combining information represented by the parameter concept defined above, a specification of a desired service or content, and a maximum number of alternatives to be assessed, our DDM approach is equipped with every input needed to calculate a deployment decision in accordance to user preferences and needs as demanded in section 1.1.3.

Figure 3.1: Class diagrams of parameters in a DDM system.
3.3 The DDM algorithm

To investigate and solve the deployment decision making problem stated in section 1.1.3, we developed an algorithm. This algorithm can be applied each time, one out of many alternatives to obtain a desired service or content has to be selected. All terms used correspond to the terminology defined in section 3.1. Information that needs to be processed is represented as explained in section 3.2. Figure 3.2 summarises all steps of the DDM algorithm.

<table>
<thead>
<tr>
<th>Algorithm – DDM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong>: Set of discovered deployment / retrieval alternatives including their service parameters</td>
</tr>
<tr>
<td>1. Select a set of adaptable and decision parameters</td>
</tr>
<tr>
<td>2. Calculate usefulness of each alternative</td>
</tr>
<tr>
<td>3. Select alternative with maximum usefulness</td>
</tr>
<tr>
<td><em>(Deploy selected alternative as consequence of DDM algorithm)</em></td>
</tr>
<tr>
<td>4. Measure decision parameter values during deployment / retrieval</td>
</tr>
<tr>
<td>5. Adapt adaptable parameters</td>
</tr>
</tbody>
</table>

The following paragraphs explain in detail all steps the DDM algorithm performs.

- **Input – a set of discovered alternatives.** A user initiates a deployment decision making process by instructing our DDM approach to obtain a special service or content. To find providers that offer the desired service or content, a service discovery mechanism can be applied. This mechanism returns a set of alternatives how to get the service or content. Each of these alternatives comprises information about its source, i.e., provider, and about the conditions under which it can be obtained. For each alternative, all conditions a provider considers to be relevant are denoted in terms of service parameters as described in section 3.2.3.
Service discovery itself is out of the scope of our research. We take a set of discovered and described alternatives as input for DDM. At the same time, we assume that the user limited the size of this set to the maximum number of alternatives that he or she wants to be assessed within the concerned decision making process.

- **Step 1 – select a set of adaptable and decision parameters.** One of the main tasks we developed our DDM approach for, is to enable finding retrieval or deployment alternatives in accordance with user preferences. This first step of our algorithm provides a facility to configure and to customise our DDM approach. Adaptable and decision parameters are selected by the user. They embody aspects that are considered relevant for a DDM process. Selected sets of parameters represent firstly, individual interests of different DDM users, and secondly, one user’s interests concerning different retrieval or deployment processes. Once a set of parameters has been composed, it can either be used for multiple DDM processes, or it can be adapted, if required, in the first step of the next DDM process that takes place. In addition to customisation, the usage of adaptable parameters enables self-adaptation by learning from observed experiences. Detailed explanations on that can be found in section 5.2. Moreover, parameter selection influences the quality of a deployment decision as well as the computational effort that needs to be spent to run the DDM algorithm. Involving more parameters leads to a better representation of user interests and alternative properties. At the same time, it causes an increased number of calculations that have to be carried out. A comprehensive investigation of the relation between number of parameters and algorithmic complexity, and the relation between number of parameters and quality of deployment decisions is presented in section 4.3. Furthermore, we introduce in that section a number of parameter selection methods which can be applied to fulfil diverse purposes. The result of step 1 of the DDM algorithm is a set of adaptable and decision parameters which are denoted in the way described in section 3.2.3, and which represent the preferences of the user of the DDM algorithm within a current DDM process.

- **Step 2 – calculate the usefulness of each alternative.** The usefulness of an alternative is a measure that expresses how well the concerned alternative corresponds to a set of adaptable and decision
parameters. These parameters have to be defined before the usefulness calculation takes place. Adaptable and decision parameters represent aspects that matter for a DDM user. Each user can define their own parameters as well as individual interpretations of the usefulness of their value. Due to that fact, the value of an adaptable or decision parameter can be any symbolic or sub symbolic information. To be able to compare different alternatives based on this arbitrary information we equipped adaptable and decision parameters with so called interpreters. Each parameter has one interpreter which can be exchanged on demand by a user. A user can define one or more individual interpreters for each parameter according to any personal preferences. An interpreter transforms the value of a parameter into an interpreted value by mapping it to a usefulness scale $S_N$. This scale is defined as follows:

**Definition 3.1: Usefulness scale.**

In a deployment decision making system, the usefulness scale $S_N$ is a measure for assessing a value $v_p$ of an adaptable or decision parameter $p$, and mapping $v_p$ to an interpreted value $i v_p$.

- $S_N = [-\infty, 1]$, $i v_p (v_p) \in S_N$ with
- $0 < i v_p \leq 1$, the parameter $p$ has the usefulness $i v_p$ for the associated deployment alternative
- $i v_p = 0$, the parameter $p$ has no relevance for assessing the usefulness of the associated deployment alternative and will not be regarded in the deployment decision making process
- $i v_p < 0$, the associated deployment alternative is not suitable for deployment because of the value $v_p$ of the parameter $p$

The overall usefulness $u_a$ of an alternative $a$ is determined by calculating the mean usefulness of all adaptable and decision parameters involved in the regarded DDM process. For that purpose, we multiply each interpreted value $i v_p$ with a weight factor
$w_p$, sum theses products up and divide them by the number of involved parameters $\#p$. Equation shows this calculation.

$$u_a = \begin{cases} 
0 & \text{if } \forall p \in (DP \cup AP), iv_p = 0 \\
-1 & \text{if } \exists p \in (DP \cup AP), iv_p < 0 \\
\sum_{p \in (DP \cup AP)} iv_p \cdot w_p & \text{for } \forall (p \in (DP \cup AP), \text{where } iv_p > 0) \text{else} 
\end{cases}$$

(3.6)

$u_a$: usefulness of alternative $a$
p: decision or adaptable parameter
$\#p$: number of involved parameters with $iv_p > 0$, $\#p > 0$
$iv_p$: interpreted value of $p$
w: weight of $p$
$DP$: set of decision parameters involved in current decision
$AP$: set of adaptable parameters involved in current decision

Computing a mean value based on the weighted sum of components is a well known practice. In computational intelligence, for example, the integration functions of artificial neural networks [94] and the affinity functions of artificial immune systems [95] are calculated in a similar manner. In the context of DDM, the function denoted in (3.6) allows us to combine a variable number of parameters where the weight factors are used to define priorities of these parameters in relation to each other. Combining different parameters is possible because we solely rely on their interpreted values which are all within the range of $S_N$. An interpreted value represents how useful any value of an adaptable or decision parameter is for the user, because the user his- or herself defined the mapping from parameter value to interpreted value according to his or her individual demands. The result of step 2 of the DDM algorithm is a rating where one overall usefulness value is created for each alternative involved in the decision making process.
• **Step 3 – select the alternative with maximum usefulness.** In this step of the DDM algorithm we figure out the alternative with the highest usefulness value compared to all others, and declare it to be the winner of the DDM process. The most efficient way to determine this winner is to memorise the alternative with the current maximum usefulness throughout step 2 of the algorithm. For this purpose, we store the current maximum usefulness and a reference to the corresponding alternative. Each time a usefulness value is calculated for an alternative, this value is compared to the current maximum usefulness. If the recent usefulness exceeds the current maximum, both, the maximum usefulness as well as the reference to the corresponding alternative are updated. The result of step 3 of our DDM algorithm is a reference to the alternative with the highest usefulness value.

• **Deploy the selected alternative.** This action is not part of the deployment decision making algorithm as such. It rather is the consequence of DDM. To perform a service deployment or content retrieval process, our DDM approach provides a reference to the alternative that was selected as winner in step 3. This reference is denoted in the same way as the input that was passed to DDM before the decision making process started. However, deploying the service or retrieving the content that is associated with the selected alternative is beyond the scope of our research work (see section 3.1). We assume this is done by a facility appropriate for the domain in which DDM is applied. The result of this action is that the deployment of the service or retrieval of the content represented by the winning alternative was initiated and took place. We do not require that this process finishes successfully.

• **Step 4 – measure decision parameter values.** Our DDM algorithm assesses alternatives based on service parameter values. These values, representing conditions to obtain the concerned alternative, are denoted by service or content providers. After a deployment or retrieval process, a DDM user can examine whether denoted service parameter values reflect reality. For this purpose, the values of the decision parameters, used to find the winning alternative in the preceding DDM process, have to be measured. How exactly this measurement needs to be done depends, firstly, on the concerned parameters, and secondly, on the underlying technology utilised for
deployment or retrieval. We do not concentrate on this issue within our research. To keep our approach as general as possible, we assume that all values that can be measured are inserted into the measured value fields of the concerned decision parameters. In this context, not necessarily all decision parameters involved in the preceding DDM process need to have measured values. If there is no measurement for a parameter, this means that no according experience was gained. The result of step 4 of the DDM algorithm is a set of measured values which represent experiences gained from the recent deployment or retrieval process. These experiences are related to the deployed service or the retrieved content, and to the provider that offered the winning alternative.

**Step 5 – adapt the involved adaptable parameters.** Adaptation of adaptable parameters is learning from experiences. Within our DDM approach, this is put into practice by the interpreters of the concerned parameters. For the fact that interpreters can be designed and exchanged by users individually (see section 3.2.3), any – current and future – learning methods can be incorporated into DDM as long as they can be implemented as an interpreter and the information available for learning is sufficient. In order to learn, interpreters can for example compare denoted service parameter values to actually measured values of corresponding decision parameters. In addition, learning can take place without the need of a comparison by counting information like how often one and the same provider was chosen. Furthermore, already learned information can be incorporated in the learning process. For instance, a history to detect trends in the development of service parameter values can be implemented this way. This step of the DDM algorithm puts into practice self-adaptation of our approach (see section 5.2). It enables reacting to changes in the environment where DDM is applied, as well as to unreliability of information. Consequently, the degree of autonomy increases because user intervention can be avoided in various situations. The result of step 5 of our algorithm is experience gained during the recent DDM process which is incorporated into experience that has already been learned before.

Applying the algorithm described above leads to calculating one deployment decision. It can be repeated any time, a user wants to
automatically decide which alternative out of many to deploy a special service or to retrieve a desired content fits best to his or her preferences and needs. After clarifying the steps that have to be performed to come to a deployment decision, we show in section 3.4 the results of an evaluation of the usefulness calculation.

3.4 Evaluation of the usefulness calculation

To justify the relevance of the problem described in section 1.1.3, and to show that the algorithm given in section 3.3 solves this problem, we state the following two assertions.

Assertion 3.1

DDM selects an alternative with maximum usefulness according to a set of user preferences.

Assertion 3.2

Applying DDM can save resources and can therefore optimise the process of deploying services.

In order to investigate the behaviour of the DDM algorithm and to find out if Assertion 3.1 and Assertion 3.2 hold, we simulated a music download scenario similar to the one presented in section 1.1.2. Aspiring maximum conciseness, the adaptations and settings described in section 3.4.1 have been applied (see also [96]).

3.4.1 Experiment 1 – selecting maximum usefulness

Jane applies a DDM system to automatically decide upon alternatives to download music files from the internet. The song, i.e., content she desires at the moment is John Lennon’s “Imagine”. The scenario comprises one target device which is the computer of Jane. Three source devices, which are operated by music providers in the internet, offer the desired content. Jane selects the following two decision parameters that have to be considered when calculating a content retrieval decision.

1. The price $p$ to obtain the desired music file; Jane prefers alternatives with a low price to those with a high price. After all,
she does not want to pay more than 5 Euros. Equation (3.7) shows Jane’s interpretation of the usefulness $u_p$ of the decision parameter price within the given scenario.

$$u_p = \begin{cases} 
1 & \text{for price } p \leq 0 \text{ Euro} \\
\frac{-p}{5} + 1 & \text{for price } 0 < p < 5 \text{ Euro} \\
-0.1 & \text{for price } p \geq 5 \text{ Euro}
\end{cases}$$  

(3.7)

2. The sampling bit rate $b$ used to digitalise the music; Jane likes most music files that are sampled with a bit rate of 192 kBit/s. A lower bit rate means too low quality and a higher one leads to unnecessary memory and CPU consumption. However, Jane neither wants to fall below a minimum of 64 kBit/s nor does she want to exceed a maximum of 320 kBit/s. Her interpretation of the usefulness $u_b$ of the decision parameter sampling bit rate is denoted in (3.8).

$$u_b = \begin{cases} 
-0.1 & \text{for bit rate } b < 64 \text{ kBit/s} \\
\max\left(0.01, \sin\left(\frac{\pi \cdot (b - 64)}{320 - 64}\right)\right) & \text{for bit rate } 64 \leq b \leq 320 \text{ kBit/s} \\
-0.1 & \text{for bit rate } b > 320 \text{ kBit/s}
\end{cases}$$  

(3.8)

Figure 3.3 and Figure 3.4 illustrate the relation between value and usefulness of the concerned parameters. By superposing the usefulness of both, price and bit rate, we obtain the search space that contains all considered alternatives. Their overall usefulness values are points on the surface spanned by the applied decision parameters. The DDM algorithm has to find the alternative with the highest usefulness compared to all other alternatives.
An algorithmic approach to service and content deployment decision making

usefulness on the scale $S_N$

Figure 3.3: Usefulness of decision parameter (a) price and (b) bit rate.

Figure 3.4: Overall usefulness of decision parameters price and bit rate.
We repeated the DDM process over ten rounds, i.e., ten times. In each round, we initialised all values of all service parameters of all three alternatives with random values. That means, we applied a generator which created pseudo-random real numbers within the range from 0.00 to 5.00 for the parameter price, and a set of bit rates including 64, 128, 192, 256, and 320 kBit/s for the parameter bit rate. Within the value ranges of price and bit rate, we utilised a uniform probability distribution. The weight of each decision parameter was set to 1 so that both, price and bit rate are considered to be equally important to Jane. All other settings were equal throughout the ten rounds. The decision parameters applied in this experiment are exemplarily chosen. As mentioned in section 3.2.3, each user can define and use an arbitrary number of individual decision parameters as well as associated interpreters. We carried out the described experiment utilising a simulation environment which we developed to investigate the behaviour of the DDM algorithm.

Figure 3.5: All deployment alternatives of ten rounds.

Figure 3.5 depicts the results of applying the DDM algorithm to the scenario described above. On the x-axis we map the number of the simulated round, i.e., 1 to 10. The y-axis shows the overall usefulness \( u \) of an assessed alternative on the scale \( S_N \). According to (3.6) and given the scenario settings, \( u \) is calculated as follows:

\[
u = \frac{u_p + u_b}{2}.
\]
The black bars represent the usefulness of alternatives that have been selected as winner by the DDM algorithm. Grey and white bars represent the alternatives that have not been selected. It can be observed that the selected alternatives always have maximum usefulness compared to the alternatives not selected within each round. These results are not surprising because we designed the DDM algorithm accordingly. What is as interesting as simple, is the consequence: Following the definition of decision parameters (see section 3.2.3) and the assumption that a user is capable of expressing which aspects matter in which way (see section 3.2.2), we can conclude that the DDM algorithm chooses the alternative that fits best to the preferences specified by the user in any of the simulated cases. This fact confirms Assertion 3.1 in the context of the simulated scenario.

Considering all alternatives throughout all ten repetitions of the DDM process, we calculated a mean usefulness of $u_{\text{mean}}=0.67$, and a minimum usefulness of $u_{\text{min}}=0.44$. If we assume not having applied DDM but choosing alternatives randomly instead, we would probably have achieved $u_{\text{mean}}$. If we assume the worst case that could have happened without applying DDM but choosing alternatives randomly instead, we would have achieved $u_{\text{min}}$. The worst case here means always selecting the alternative with minimum usefulness throughout all ten rounds. In contrast, selecting alternatives by utilising DDM throughout all ten rounds led to achieving a mean usefulness of $u_{\text{selected}}=0.83$. These numbers underpin that Assertion 3.1 holds within the context of the performed experiment.

### 3.4.2 Experiment 2 – saving resources

In a second experiment, Jane wants to save computational resources in three respects. Firstly, she tries to minimise the amount of data that needs to be transferred when retrieving content. Secondly, Jane requires saving disc space. Thirdly, she likes to avoid causing heavy processor load when playing obtained music on her device. For that purpose, Jane prefers music files that are sampled with a bit rate of only 64 kBit/s. Consequently, (3.10) is derived from (3.8) in the following manner:
All other settings remain as defined in section 3.4.1. Service parameters are initialised with exactly the same values as in the former experiment. Figure 3.6 illustrates the adapted relation between value and usefulness of the parameters price and bit rate. Table 3.1 summarises usefulness and bit rate values.

Figure 3.6: Overall usefulness of decision parameters *price* and *bit rate*. 

\[ u_b = \begin{cases} 
-0.1 & \text{for bit rate } b < 64 \text{ kBit/s} \\
\max \left( 0.01, \sin \left( \frac{\pi \cdot (b + 64)}{320 - 64} \right) \right) & \text{for bit rate } 64 \leq b \leq 320 \text{ kBit/s} \\
-0.1 & \text{for bit rate } b > 320 \text{ kBit/s} 
\end{cases} \] (3.10)
An algorithmic approach to service and content deployment decision making

<table>
<thead>
<tr>
<th></th>
<th>Extreme value</th>
<th>Mean value</th>
<th>Selected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usefulness (on the scale $S_N$)</td>
<td>$u_{\text{min}}=0.30$</td>
<td>$u_{\text{mean}}=0.50$</td>
<td>$u_{\text{selected}}=0.72$</td>
</tr>
<tr>
<td>Bit rate (in kBit/s)</td>
<td>$b_{\text{max}}=236.80$</td>
<td>$b_{\text{mean}}=177.07$</td>
<td>$b_{\text{selected}}=108.80$</td>
</tr>
</tbody>
</table>

Table 3.1: Overview of usefulness and bit rate values.

In terms of usefulness, the results of the experiment are similar to the results presented in section 3.4.1. In each round, the alternative with maximum usefulness was selected as winner by the DDM algorithm. Considering all ten rounds, we observed an average round minimum for the usefulness of $u_{\text{min}}=0.30$ on the scale $S_N$, and an overall mean usefulness of $u_{\text{mean}}=0.50$ on the scale $S_N$. These worst and average cases that might have had occurred without applying DDM, are outraged by an average usefulness of $u_{\text{selected}}=0.72$ on the scale $S_N$ for the alternatives selected with the help of the DDM algorithm. If we analyse the parameter bit rate, the experiment resulted in an average round maximum of $b_{\text{max}}=236.80$ kBit/s which is the worst case. The overall mean bit rate is $b_{\text{mean}}=177.07$ kBit/s which is the average case. The average bit rate Jane got by retrieving the alternatives selected by DDM is $b_{\text{selected}}=108.80$ kBit/s. These numbers imply utilising the DDM algorithm saves computational resources by preferring music files with lower bit rates, if this is in line with the preferences of the DDM user. Consequently, Assertion 3.2 is confirmed within the context of the performed experiment.

3.4.3 Experiment 3 – extended music download scenario

We repeated the experiment described in sections 3.4.1 and 3.4.2 a third time (see also [97]). To explore how the DDM algorithm operates in more complex scenarios, we now applied 20 different deployment alternatives. Furthermore, we aimed at achieving statistically more reliable statements by increasing the number of rounds, i.e., repetitions to 50. As decision parameters we again used the price and the sampling bit rate (with a maximum usefulness at 192 kBit/s, cf. section 3.4.1). All other settings remained unchanged.
Underpinning the preceding experiments, Figure 3.7 shows that the usefulness of the alternatives selected by DDM is always higher than the minimum and mean usefulness regarding all alternatives of a simulation round. In this experiment we calculated an average round minimum usefulness of $u_{\text{min}}=0.16$ on the scale $S_N$, an overall mean usefulness of $u_{\text{mean}}=0.57$ on the scale $S_N$. The average usefulness of all alternatives that won a DDM process is $u_{\text{selected}}=0.92$ on the scale $S_N$.

Based on the experimental results, we can conclude that the DDM algorithm was able to handle the scenario comprising 20 alternatives, and that alternatives were selected that are in line with user preferences specified by a set of decision parameters. Following the findings of the experiment described in section 3.4, it is clear that the DDM approach proposed in this thesis has the potential of optimising the resource consumption of service and content related computing networks, as well as satisfying user demands.

![Figure 3.7: Minimum, mean, and selected usefulness of 20 alternatives for 50 rounds.](image)

### 3.5 Summary

In section 3, we elaborated on the details of the DDM process. After defining a terminology that explains all basic elements and processes, we delimited our work from adjacent topics. While concentrating on how to come to service deployment or content retrieval decisions, we do neither focus on the discovery of deployment or retrieval alternatives, nor do we
address deployment or retrieval processes themselves. To deepen the understanding of the knowledge we need to come to deployment decisions, we characterised the nature of the information we are faced with in the defined computing network domain. Subsequently, we listed which information is required, and which is available for DDM. To represent the concerned information, we introduced a parameter concept comprising service parameters, decision parameters, and adaptable parameters. The first, service parameters stand for conditions and properties of deployment or retrieval alternatives. The second, decision parameters correspond to preferences and needs of a user applying DDM. And the third, adaptable parameters symbolise experiences learned. Furthermore, we defined common parameters which enable assessing multiple alternatives upon an equal base of information. The fact that parameters can be defined and exchanged according to user needs and environmental demands enables flexible application of DDM as demanded in section 1.1.3.

After analysing informational requirements, we proposed an algorithm for making deployment or retrieval decisions, and we explained in detail each of its steps. Two assertions have been stated to justify the relevance of the proposed algorithm in order to solve the DDM problem. Firstly, we stated that DDM selects an alternative with maximum usefulness according to a set of user preferences. Secondly, we claimed that applying DDM can save computational resources. Three experiments confirmed both assertions within the context of a music download scenario similar to the one described in section 1.1.2.
4 Algorithmic complexity analysis

In section 3 we have introduced an algorithm for solving the deployment decision making problem. With a set of experiments we showed that this algorithm works correct. That means that it selects one alternative out of many which fits best to the preferences of a DDM user. However, we want our DDM approach to be applicable in large scale scenarios where several hundred of alternatives may exist, and choices are made based on dozens of decision and adaptable parameters. For that fact we have to investigate: Does DDM work efficiently? Efficiently means that our algorithm achieves correct results while consuming a reasonable amount of computational resources. This condition has to hold despite a growing size of input data. To gain insight into the behaviour of DDM, we analysed the relation between kind and size of input data, and performance of the algorithm. We investigated which factors influence a DDM process. For that reason we carried out an algorithmic complexity analysis and describe our findings within the following sections. Firstly, we clarify necessary fundamentals and the notation applied. Next, the complexity of each step of the DDM algorithm is analysed. And finally, different methods to select parameter subsets are presented to illustrate in which way the performance of DDM can be influenced.

4.1 Fundamentals and notation

It is well known that computational problems in computer science are usually solved by creating an appropriate algorithm. According to Wegener, we can formalise that such computational, i.e., “algorithmic problems include all problems that can be handled by computers and for which we can unambiguously distinguish between correct and incorrect solutions. Among these are optimisation problems and problems with unique solutions such as evaluation problems and decision problems [...]” [98]. The meaning of the word “algorithm” depends on time and context where it is used [99]. Nevertheless, in computer science of the 21st century, an algorithm is commonly understood to be a well-defined sequence of computational steps that transform an input value into an output value [100].
In order to decide if an algorithm, developed to solve a certain problem can be applied to a dedicated domain, the question has to be clarified how many computational resources the algorithm consumes. Those *computational resources* comprise, among others, accessing hardware components (like printers or hard disks), traffic to and from a connected network, and the utilisation of CPUs. Once the input for the DDM algorithm is available, we do not utilise resources other than CPU and memory. Furthermore, we assume that computationally limited devices like smartphones are equipped with several Gigabytes of memory. This magnitude is considered to be sufficient for the scenarios described in sections 1.1.1 and 1.1.2, as well as for potential fields of application. For these reasons, we suppose that the most significant criterion to assess the applicability of our algorithm in terms of algorithmic complexity is the speed of calculating a deployment or retrieval decision, which we refer to as the algorithm’s *runtime*.

For evaluating the resource consumption of an algorithm, we utilise the principles of *complexity theory*, which is common practice in computer science. Wegener argues that “the goal of complexity theory is to prove for important problems that their solutions require certain minimum resources. The results of complexity theory have specific implications for the development of algorithms for practical applications” [98]. We are especially interested in determining the amount of resources, i.e., time, DDM consumes if input data of a limited length causes the most complicated and thus expensive calculations. We call this the worst case *runtime* or upper bound of the DDM algorithm depending on the length of the input data. Once we have found an upper bound, it is clear that DDM will not consume more resources than this upper bound. If this amount of resources is reasonable with regard to the length of the input data, we can say that DDM works efficiently. Further emphasis concerning the complexity of DDM is put on the algorithm’s *average case runtime* which is defined as the mean runtime of an algorithm $A$ for a probability distribution $q_n$ on the inputs of length $n$ [98]. We use this notation to assess the performance gain achieved by a tuning mechanism we developed for DDM (see section 4.4).

For expressing our complexity considerations, the Landau-notation is used, which is a formalism to describe asymptotic growth of the runtime of algorithms depending on the length of the input data. To be concise, we only explain the concerning parts of this notation we used in this work. More detailed information can be found in [98][99][101]. Due to the fact
that the Landau-notation is a mathematical concept, it is related to any kind of functions. However, in the context of computer science it is often applied to algorithms. That means, what is expressed when using Landau-notation, especially in this thesis, is the runtime of an algorithm as a function of the length of the algorithm’s input data. As argued above, we concentrate on deducing an asymptotic upper bound for the runtime of the DDM algorithm which is denoted by the O-notation. Consequently we will only mention, but not define, that asymptotically tight bounds are denoted using Θ-notation and asymptotically lower bounds are denoted using Ω-notation. According to Cormen, Leiserson, Rivest, and Stein [100] O(g(n)) is defined as follows.

**Definition 4.1 (Cormen, Leiserson, Rivest, and Stein): O-notation.**

Let n be a natural number $n \in \mathbb{N} = \{0, 1, 2, ...\}$.
For a given function $g(n)$, we denote by $O(g(n))$ the set of functions
$O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0 \}$.

In other words, Definition 4.1 implies that for sufficiently large constants $n$ and $c$, $f(n)$ grows not faster than $c \cdot g(n)$. Complexity theory uses a set of classes to separate problems with different orders of growth. Even though such classes exist in terms of time and space consumption, we utilise the runtime exclusively. The performance of an algorithm is commonly accepted as efficient, if the algorithm’s runtime complexity is polynomial. Table 4.1 provides an overview of the most common complexity classes. The degree of efficiency of an algorithm decreases as the order of growth of runtime increases.
A function $f(n)$ is called:

<table>
<thead>
<tr>
<th>Complexity class</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>$f(n) = O(1)$</td>
</tr>
<tr>
<td>logarithmic</td>
<td>$f(n) = O(\log^k n)$ for some $k \in \mathbb{N}$</td>
</tr>
<tr>
<td>linear</td>
<td>$f(n) = O(n)$</td>
</tr>
<tr>
<td>quasi-linear</td>
<td>$f(n) = O(n \cdot \log^k n)$ for some $k \in \mathbb{N}$</td>
</tr>
<tr>
<td>quadratic</td>
<td>$f(n) = O(n^2)$</td>
</tr>
<tr>
<td>cubic</td>
<td>$f(n) = O(n^3)$</td>
</tr>
<tr>
<td>polynomial</td>
<td>$f(n) = O(n^k)$ for some $k \in \mathbb{N}$</td>
</tr>
<tr>
<td>exponential</td>
<td>$f(n) = \Omega(2^{n^\varepsilon})$ for some $\varepsilon &gt; 0$</td>
</tr>
</tbody>
</table>

Table 4.1: Common complexity classes.

Now that we clarified how to measure and classify the runtime of an algorithm, we need to define the unit of measurement that we apply. Counting runtime in terms of seconds implies that one and the same algorithm runs differently long on two computers with dissimilar computational power. This implication holds for every two divergent computing devices. Consequently, general statements concerning an algorithm’s performance are inappropriately difficult which means that counting time is not a suitable cost model. Various other methods for measuring resource consumption have been developed within the scope of complexity theory. Each pays attention to aspects like the number of computational steps, different expense of arithmetic operations, or extent of processed numbers [98]. As widely applied in computer science for similar considerations, we measure the complexity of DDM depending only on the algorithm itself and on the input data. This enables us to judge the performance of algorithms independent from technical details like type and architecture of the computer running an algorithm. Furthermore, the costs of algorithms assessed in this way can be translated to any future computers easily. Counting only computational operations that depend on a set of predefined relevant variables, i.e., operands, but not on the size of any operant, is referred to as uniform cost model [98]. For assessing the performance of the DDM algorithm, we consider the variables denoted in Table 4.2 as relevant. Additionally, we denote the number of common parameters (see section 3.2.3) by writing $\#cp$ where $\#cp \leq \#dp$. All decision
parameters that are selected in the first step of the DDM algorithm are summarised with the symbol \( \#ps \) where \( \#ps \leq \#dp \).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>#a</td>
<td>Number of alternatives from which the most useful one is selected</td>
</tr>
<tr>
<td>#dp</td>
<td>Number of available decision parameters</td>
</tr>
<tr>
<td>#ap</td>
<td>Number of available adaptable parameters</td>
</tr>
<tr>
<td>#spa</td>
<td>Number of service parameters of an alternative ( a )</td>
</tr>
</tbody>
</table>

Table 4.2: Variables relevant for assessing the runtime performance of the DDM algorithm.

### 4.2 Algorithmic complexity of DDM

Within the following paragraphs we deduce step by step the complexity of the DDM algorithm presented in section 3.3. By doing so, we adhere to the notation and conventions made in section 4.1. The result of this analysis will be an understanding of how the DDM algorithm’s performance is affected by which factors (see also [102]).

**Input.** The input of the DDM algorithm comprises a number of alternatives \( \#a \) to deploy a desired service or retrieve desired content. Each alternative \( a \) is described by \( \#sp_a \) service parameters. For assessing the alternatives, \( \#dp \) decision parameters and \( \#ap \) adaptable parameters are applied. Following that, we obtain \( \#a, \#sp_a, \#dp, \) and \( \#ap \) as relevant variables as denoted in Table 4.2.

**Step 1** – selecting a set of adaptable and decision parameters – can be achieved in two ways. Firstly, all decision and adaptable parameters can be considered, which implies that alternatives might be assessed on a dissimilar base of information (see section 3.2.3). Secondly, all adaptable, but only common decision parameters (i.e., decision parameters that have a corresponding service parameter for each assessed alternative) can be regarded. The consequences of this procedure are an equal base of assessment, but an additional effort that needs to be spent to determine which parameters are common. Whether using all or only common parameters, step 1 comprises finding matching service parameters for as
An algorithmic approach to service and content deployment decision making

many decision parameters as possible, considering each alternative. Assuming the worst case, we have to take each alternative, for each alternative take each decision parameter, and for each decision parameter, test each service parameter whether it matches or not. Consequently, selecting a subset of all decision and adaptable parameters and finding matching service parameters for each alternative consumes the following costs $c_{1a}$:

$$c_{1a} \leq \# a \cdot \# dp \cdot \# s_p a + \# a \cdot p.$$  \hspace{1cm} (4.1)

When selecting a subset of all adaptable parameters but only common decision parameters, we have to consider, first, the additional effort for determining which decision parameters are common, and second, the fact that $\# c_p \leq \# d_p$. Regarding this and finding matching service parameters for each alternative, step 1 consumes the following costs $c_{1c}$:

$$c_{1c} \leq \# a \cdot \# s_p a \cdot \# dp + \# a \cdot \# s_p a \cdot \# c_p \cdot \# a p.$$  \hspace{1cm} (4.2)

**Step 2** – calculating the usefulness of each alternative – involves mapping the values of all selected decision parameters ($\# p s$) and adaptable parameters selected in step 1 to the usefulness scale $S_N$ (see Definition 3.1). Since this mapping, i.e., interpretation does not depend on the relevant variables listed in Table 4.2 we count each interpretation with constant costs. Therefore, the following costs $c_2$ have to be spent for step 2:

$$c_2 \leq \# a \cdot (\# p s + \# a p).$$  \hspace{1cm} (4.3)

**Step 3** – selecting the alternative with maximum usefulness – can be performed with constant costs $c_3$:

$$c_3 \leq \text{const.}$$  \hspace{1cm} (4.4)

This is possible because we can carry along the alternative with maximum usefulness throughout step 2.

**Deploying the selected alternative** is not part of the DDM algorithm itself. For that fact, we do not need to count any costs.

**Step 4** – measuring decision parameter values – is performed during the deployment or retrieval of the selected alternative. Because only the alternative with maximum usefulness will be deployed or retrieved, the
values of all selected decision parameters of solely this alternative can be measured. The resulting costs \( c_4 \) are:

\[
c_4 \leq \# ps.
\]  

**Step 5** – adapting the involved adaptable parameters – comprises handing over all information about denoted and actually measured service and decision parameter values of the concerned DDM process to the interpreter of each involved adaptable parameter. Let us assume that this information was stored in a proper data structure for all alternatives during step 1, where matches between selected decision and service parameters have been determined. Given such a data structure, the costs of step 5 arise from calling an adaptation method of each selected adaptable parameter. Following that, we can estimate the costs \( c_{5e} \) as follows:

\[
c_{5e} \approx \# ap.
\]  

Nevertheless, (4.6) does not represent the worst case. The size of the information passed depends on the number of alternatives, selected adaptable and decision parameters, and service parameters of each alternative that is assessed. For all these variables are considered to be relevant (see Table 4.2), we have to regard them when counting the costs of step 5. The way of interpreting these variables depends on the implementations of the concerned adaptable parameters. Since we do not know how all individual adaptable parameters will be implemented in the future, we are not able to further restrict the costs \( c_5(\#a, \#dp, \#ap, \#sp_a) \) of this step. We have to use them as a function instead as denoted in (4.7).

\[
c_5 = f(\#a, \#dp, \#ap, \#sp_a).
\]  

Summarising all steps of the DDM algorithm by combining (4.2)–(4.7), the overall costs \( c_{DDM} \) are:

\[
c_{DDM} \leq \# a \cdot \# sp_a \cdot \# dp + \# a \cdot \# sp_a \cdot \# cp + \# ap
\]
\[+ \# a \cdot (\# ps + \# ap) + \text{const} + \# ps + c_5(\# a, \# dp, \# ap, \# sp_a).
\]
With regard to the facts that $\# ps \leq \# dp$, $\# cp \leq \# dp$, and expressed in terms of asymptotical runtime growth concerning the relevant variables denoted in Table 4.2, an upper bound for the costs of DDM is:

$$c_{DDM} = O(\# a \cdot (\# sp_a \cdot \# dp + \# ap) + c_5(\# a, \# dp, \# ap, \# sp_a)).$$  \hspace{1cm} (4.9)$$

From (4.7) it can be concluded that the complexity of step 5 depends on the implementation of interpreters utilised for adaptable parameters. This implementation can be individually done by DDM users. Consequently, users can decide upon the amount of computational resources to spend at this point of the algorithm. Step 5 is of polynomial expense, i.e., works efficiently if the user implementation of all interpreters of each involved adaptable parameter is efficient. For this fact, the DDM algorithm proposed in this thesis is of polynomial expense, which means that it works efficiently if $c_5$ is polynomial.

### 4.3 Influencing the algorithmic complexity

In the succeeding paragraphs, we describe and evaluate how the computational costs to calculate a deployment or retrieval decision can be influenced. We therefore introduce several parameter selection methods and deduce their properties and effects within our DDM approach.

#### 4.3.1 Effects of parameter selection for DDM

Beside the actual algorithmic complexity of the DDM algorithm, another finding results from the analysis described in section 4.2. When we study (4.8) and (4.9), it becomes obvious that influencing the relevant variables that are taken as input for DDM also affects the computational effort needed to come to a deployment or retrieval decision. Deconstructing (4.8), we can conclude that, at the moment, we cannot manipulate $\# sp_a$. Which and how many decision parameters are made available depends on the providers of services and content. However, a service parameter selection mechanism might be developed and investigated in the scope of future work (see section 6.3). How to choose a subset of all available
service parameters, how to do this for each alternative, and how this affects the decision making process are research questions that might be answered in succeeding investigations. An appropriate limit for the number of alternatives that have to be assessed during a DDM process is assumed to be part of the input data (see section 3.3). For that fact, no additional control mechanism is needed for this task. Our DDM approach enables users to choose for each decision process subsets of all available decision and adaptable parameters. In the following, we explain our investigation of effects that occur when subsets of decision parameters interact with corresponding service parameters, preluding with Assertion 4.1.

**Assertion 4.1**

*Selecting decision parameters for a DDM process is a trade-off between spending computational resources and getting the most useful deployment or retrieval decision.*

Based on (4.8), it is obvious that reducing the number of involved decision parameters directly leads to a reduction of computational resources needed to come to a deployment or retrieval decision. To show that Assertion 4.1 holds, we need to derive that applying less decision parameters may cause a loss in the quality of the concerned decision in terms of usefulness.

### 4.3.2 Parameter selection methods

Due to the fact that each DDM user may pursue individual goals, it can be said that one optimal decision parameter selection method does not exist. The question that rather has to be answered is which selection method is most suitable for which purpose. In the following paragraphs we denote several methods and deduce their properties as well as their influence on the DDM algorithm.

**4.3.2.1 Selecting all decision parameters**

If a computing system where DDM is running is sufficiently equipped with computational resources, the DDM user might want to calculate the most useful deployment decision. For that purpose, all available information
An algorithmic approach to service and content deployment decision making

concerning all involved deployment alternatives, i.e., all decision parameters and their corresponding service parameters have to be considered. As a consequence, no decision parameter selection is necessary beforehand. Equation (4.10) denotes the overall costs for this case. It is derived from (4.8) and rearranged in a way that shows the influence of \( \#dp \) and \( \#ap \). Furthermore, we excluded the costs for determining common parameters because this step is not required when all decision parameters are selected (insert (4.1) instead of (4.2) into (4.8)).

\[
c_{DDM\_All} \leq \#dp \cdot (\#a \cdot \#sp_a + \#a + 1) + \#ap \cdot (\#a + 1) + \text{const} + c_5(\#a, \#dp, \#ap, \#sp_a). \tag{4.10}
\]

We refer to the decision parameter selection method described in this paragraph as \texttt{selectAllMatchingDps}.

### 4.3.2.2 Selecting maximum \( x \) of all decision parameters

In an environment where the computational resource limits are well known, it can be advantageous to set a well-defined upper bound \( x \) for the number of decision parameters that are involved in a DDM process. If this method is applied, the DDM system tries to find maximum \( x \) pairs of matching decision and service parameters for each alternative. That procedure implies two facts. Firstly, it may occur that less than \( x \) pairs are found. Secondly, a different number and kind of decision parameters might be selected for each alternative. As a consequence, selecting maximum \( x \) of all decision parameters can lead to a deployment decision that is derived based on different conditions for the regarded alternatives. Nevertheless, there is a well-defined upper bound for the involved decision parameters, and so, for the expense of the DDM process, which is denoted in (4.11). In the worst case, \( x \) is equal to \( \#dp \) so that (4.10) and (4.11) do not differ.

\[
c_{DDM\_MaxXOfAll} \leq x \cdot (\#a \cdot \#sp_a + \#a + 1) + \#ap \cdot (\#a + 1) + \text{const} + c_5(\#a, x, \#ap, \#sp_a). \tag{4.11}
\]

In (4.11), the maximum number of selected decision parameters \( x \) is never greater than the overall number of decision parameters \( \#dp \). We refer
to the decision parameter selection method described in this paragraph as \textit{selectMaxXOfAllDps}.

\subsection*{4.3.2.3 Selecting all common decision parameters}

One major characteristic of the selection methods that choose all, or maximum $x$ of all decision parameters is that they do not necessarily utilise the same number and kind of decision parameters. In the case that a deployment decision should be made on exactly the same base of information for all alternatives, the concept of common parameters (see section 3.2.3) can be applied. This equal base of assessment comes at the cost of needing to spend additional effort for determining which decision parameters are common for all involved alternatives. Deduced from (4.8) while applying (4.2) instead of (4.1), the complexity for DDM utilising all common parameters is denoted in (4.12). In the worst case, in terms of algorithmic complexity, the number of common parameters $\#cp$ equals to $\#dp$. We refer to the decision parameter selection method described in this paragraph as \textit{selectAllCommonDps}.

\begin{equation}
    c_{\text{DDM}_\text{AllCommon}} \leq \#a \cdot \#sp_a \cdot \#dp + \#cp \cdot (\#a \cdot \#sp_a + \#a + 1) + \#ap \cdot (#a + 1) + \text{const} + c_s(\#a, \#cp, \#ap, \#sp_a). \label{eq:all_common}
\end{equation}

\subsection*{4.3.2.4 Selecting maximum $x$ of all common decision parameters}

When applying this method, a maximum number of $x$ decision parameters are utilised. These are selected among all common parameters, which means that all alternatives are assessed on the same base of information. If the overall number of common parameters $\#cp$ is less than $x$, than only $\#cp$ common parameters are chosen. In the worst case, $x$ equals to $\#cp$, and $\#cp$ equals to $\#dp$. Equation (4.13) shows the costs for DDM selecting maximum $x$ of all common decision parameters. We refer to the decision parameter selection method described in this paragraph as \textit{selectMaxXOfCommonDps}.
All methods mentioned above can be distinguished based on two major attributes. The first attribute is the way an upper bound for selecting decision parameters is specified. It can be defined as absolute value $x$ or not defined at all. Beside these, further kinds of upper bounds, such as a percentage of all or all common decision parameters, can be applied as well. The second attribute is the base of information upon which decision parameters are selected. The options described above are choosing all or all common decision parameters. Combinations of both variants may suit further needs.

If we assume the worst case, decision parameter selection does not reduce the expense of computational resources needed for the DDM algorithm. However, if the number of selected decision parameters is relatively small compared to the overall number of decision parameters available, and if corresponding decision – service parameter pairs are found quickly, parameter selection has a positive effect on the runtime performance of DDM. As stated in Assertion 4.1, this reduction of runtime comes at the cost of the quality of deployment decisions. Not involving all available parameters might lead to not finding the alternative with maximum usefulness.

### 4.3.3 Experimental evaluation of decision parameter selection

To investigate and understand the relation between decision parameter selection and usefulness reduction, we carried out a set of experiments. Within these experiments, we observed how the number and kind of applied decision parameters affect the deployment or retrieval decision, and the usefulness of a selected alternative of a DDM process. For that purpose, we compared all parameter selection methods described above based upon the same scenario which comprises 50 source devices that all offer one desired service. This desired service has to be deployed to one target device. As for the experiments described in section 3.4, we utilised the same simulation environment. For each parameter selection method, the experiment was repeated 50 times. All service parameter values were initialised with

\[
\begin{align*}
\text{cost}_{\text{DDM}_{\text{MaxXOfCommon}}} & \leq a \cdot \text{sp} \cdot dp + x \cdot (a \cdot \text{sp} + a + 1) \\
& + a \cdot \text{sp} \cdot (a + 1) + \text{const} + c_5(a, x, a, \text{sp}).
\end{align*}
\]
random values, i.e., using a generator which created pseudo-random real numbers $v$. All values $v$ are uniformly distributed within the range $-0.5 \leq v < 1.5$. Moreover, 50 decision parameters are applied in each round of all experiments. According to the usefulness scale $S_N$, service parameter values are always interpreted as denoted in (4.14).

$$iv(v) = \begin{cases} 
-1 & \text{for } v < 0 \\
v & \text{for } 0 \leq v \leq 1 \\
1 & \text{for } v < 1 
\end{cases}$$

(4.14)

All 50 source devices can expose 50 service parameters, where each of them is present with a probability of 0.99. Which service parameters are regarded can differ from round to round but is always equal in the same round for all investigated parameter selection methods. No adaptable parameters have been applied at all. Except number and kind of parameters that are regarded for decision making, all other settings remained unchanged throughout all 50 rounds, i.e., repetitions of the experiment. The following paragraphs explain the details of the experiments carried out.

4.3.3.1 Experiments without explicit upper bound

The parameter selection method explained in section 4.3.2.1 utilises all available decision parameters to calculate a deployment decision. That means no explicit upper bound is defined that restricts the exploitation of available information represented by decision and service parameters. Consequently, the most useful alternatives are selected which, in turn, consumes most of the computational resources. We performed an experiment where this setting is applied throughout 50 rounds. Each round corresponds to one run of this experiment. Figure 4.1 depicts that when comparing all mentioned decision parameter selection methods, based on equal settings for the concerned experiments, no usefulness value is higher than the corresponding value when all decision parameters are selected. The line connecting all separate values of $selectAllMatchingDps$ emphasises this fact.
Applying the method `selectAllCommonDps` as described in section 4.3.2.3 implies that the number of utilised parameters only depends on how many common parameters are available. This number is equal to or less than the overall number of all decision parameters. As a consequence of only using common decision parameters, not all available information might be taken into account when calculating a retrieval or deployment decision. However, the same base of judgement, which means, the same set of decision parameters is considered for assessing all involved alternatives. Figure 4.2 shows the usefulness of the selected alternative utilising all decision parameters in contrast to the usefulness of the selected alternative utilising all common decision parameters.

The experiment was repeated over 50 rounds, where in a first DDM process per round the winning alternatives were determined. A second DDM process per round was run in which the usefulness of the selected alternatives when utilising all common parameters was calculated again, using all available parameters this time. This extra step was necessary to be able to compare the usefulness values obtained applying `selectAllCommonDps` with the values obtained in the experiment applying `selectAllMatchingDps`. As depicted in Figure 4.2, the corresponding usefulness of the alternative selected when utilising all common parameters
is always equal or slightly lower than the usefulness of the alternative selected when utilising all decision parameters. The vertical variation lines emphasise this fact.

![Comparison of the usefulness when using all, and all common decision parameters.](image)

**Figure 4.2:** Comparison of the usefulness when using all, and all common decision parameters.

### 4.3.3.2 Experiments with absolute upper bound

Explicitly specifying an upper bound for the number of decision parameters regarded within a DDM process is an effective mechanism to keep control over the computational resources that are spent to determine the most suitable deployment alternative. The parameter selection method presented in section 4.3.2.2 allows defining such an upper bound $x$ based on all decision parameters. The method explained in section 4.3.2.4 enables specifying $x$ based on all common parameters. We carried out two experiments in which we compared the usefulness of selected alternatives, firstly, selecting all and maximum $x$ of all decision parameters, and secondly, selecting all and maximum $x$ of all common decision parameters. Each experiment was repeated over 50 rounds where the round number corresponds to $x$, i.e., the maximum number of decision parameters utilised.

The results of these experiments are depicted in Figure 4.3 and Figure 4.4. Both charts show that applying a parameter selection method that defines an upper bound leads to obtaining equal or lower usefulness values.
for the selected alternatives, compared to utilising all available decision parameters. A second fact that can be observed in both figures is the decreasing discrepancy between the compared usefulness values, when the number of utilised decision parameters $x$ increases. According to (4.8), achieving an increase of usefulness by applying more decision parameters requires to spend more computational resources. When comparing $\text{selectMaxXOfAllDps}$ with $\text{selectMaxXOfCommonDps}$ (see Figure 4.3 and Figure 4.4) we can observe that applying the former method leads to slightly better results in terms of usefulness of selected alternatives. Particularly, when the number of utilised decision parameters $x$ approaches the overall number of decision parameters, the usefulness values of $\text{selectMaxXOfAllDps}$ and $\text{selectAllMatchingDps}$ are becoming equal. This effect is caused by the fact that almost all available information is used for decision making when $x$ is only slightly less than 50. The reason for observing this effect in an alleviated manner when applying $\text{selectMaxXOfCommonDps}$ is that even when $x$ approaches 50, it is not granted that all decision parameters are common. Because of equipping alternatives with service parameters with a probability of 0.99 only (see experiment settings described in section 4.3.3) it is likely that the number of common parameters is less than 50. This implies that not all information is used for decision making and leads to obtaining slightly worse usefulness values compared to $\text{selectMaxXOfAllDps}$.

![Comparison of the usefulness when utilising all, and maximum $x$ of all decision parameters.](image)

**Figure 4.3:** Comparison of the usefulness when utilising all, and maximum $x$ of all decision parameters.
4.3.3.3 Findings from the experiments

The experiments described in section 4.3.3 clearly show that applying a decision parameter selection method when computing a DDM process, influences the usefulness of the calculated retrieval or deployment decision. One main result is that the usefulness of the selected alternative is always best when the number of utilised parameters is not restricted, i.e., when all available information is used.

Figure 4.2 and Figure 4.4 show that utilising common parameters results in slightly poorer usefulness values of selected alternatives than utilising any decision parameters. This is due to the fact that the number of common decision parameters is always less than or equal to the overall number of decision parameters. However, calculating deployment or retrieval decisions based on common parameters has the advantage of assessing all alternatives under equal conditions, which means, on the same base of information.

A comparison of Figure 4.3 and Figure 4.4 indicates a discrepancy between the maximum usefulness (reached when applying `selectAllMatchingDps`) and the usefulness that is obtained when only a
subset of all available decision parameters is selected (when \texttt{selectMaxXOfAllDps} or \texttt{selectMaxXOfCommonDps} is applied). This discrepancy grows as the number of involved decision parameters $x$ declines.

From the findings of our experiments we can conclude that applying more decision parameters helps to achieve higher usefulness values of the alternatives that win a DDM process. Combining this result with the theoretical deduction of the algorithmic complexity confirms Assertion 4.1. Selecting decision parameters for a DDM process is a trade-off between spending computational resources and getting the most useful deployment or retrieval decision.

To complete the described experimental evaluation, Figure 4.5 summarises the average usefulness of 50 rounds for each decision parameter selection method.

![Average usefulness of decision parameter selection methods](image)

**Figure 4.5:** Average usefulness of all decision parameter selection methods.

### 4.4 Improving the average runtime performance of DDM

In section 4 we elaborate on the algorithmic complexity of our DDM algorithm. After theoretically deriving the worst case runtime performance and evaluating facilities to influence the expense of computational resources of a DDM process, we describe in this sub section a mechanism to tune the performance of DDM considering an average case.
4.4.1 Considering maximum attainable usefulness

When analysing the algorithmic complexity of DDM in section 4.2, we referred to the worst case. That is, we assumed the input that causes maximum expense in terms of computational resources. However, the runtime of the DDM algorithm as we presented it in section 3.3 does not depend on the values of the input variables we consider to be relevant. Rather the performance is influenced by the number of these variables, and the order decision and service parameters are tried to be matched to each other.

How matches between decision and service parameters are found rely to a large extend on the order in which service parameters are presented by service and content providers. We will not consider optimising the matching process. Nevertheless, potential for further runtime improvement may exist here that can be investigated in the scope of future work (see 6.3). The number of relevant variables forming the input for DDM still needs to be kept mutable. As shown in sections 4.3.2 and 4.3.3, a restriction of the size of input data can lead to a loss of quality of calculated deployment or retrieval decisions. What we do instead is changing the DDM algorithm by introducing an additional step. During this step, we compute and evaluate the potential an alternative has to win a selection process. We call this mechanism the evaluation of maximum attainable usefulness (MAU).

Definition 4.2: Maximum attainable usefulness (MAU).

Let $a$ be an alternative in a DDM process. Let $(s-1)$ be the number of decision and adaptable parameters that have already been evaluated for $a$. So, $u(s-1)$ is the usefulness of alternative $a$ after evaluating $(s-1)$ parameters. The maximum attainable usefulness $u_{MAU}(s)$ is the usefulness $a$ can achieve if all interpreted values of all considered decision and adaptable parameters that still have to be evaluated, after $(s-1)$ parameters are already evaluated, are assumed to be 1.
An algorithmic approach to service and content deployment decision making

\[ u_{MAU}(s) = \frac{u(s-1) \cdot (s-1) + 1 \cdot (#dp+#ap - s + 1)}{#dp+#ap} . \]  

(4.15)

\( s \in \mathbb{N}, \text{ and } 0 < s \leq #dp+#ap \)

\[ u_{MAU}(0) = 0 \]

\#dp: number of considered decision parameters

\#ap: number of considered adaptable parameters

Definition 4.2 implies that \( u_{MAU}(s) \) depends on how many parameters already have been, and how many still have to be evaluated. Consequently, we have to calculate \( u_{MAU}(s) \) for each evaluation step of all involved decision and adaptable parameters, and we do this for each assessed alternative, i.e., in step 2 of the DDM algorithm. However, the costs of calculating \( u_{MAU}(s) \) do not depend on any of the defined relevant variables as such. Though, the number of involved decision and adaptable parameters appears in (4.15), it does not matter how large this number is (see definition of the uniform cost model in section 4.1). For that fact, the costs of the additionally needed step to compute \( u_{MAU}(s) \) are constant, i.e., O(1). The algorithmic complexity of the DDM algorithm remains unchanged. Figure 4.6 depicts how the calculation of maximum attainable usefulness values is performed algorithmically. The MAU algorithm replaces step 2 (calculating the usefulness of each alternative) of the DDM algorithm (see section 3.3)

**Algorithm – MAU (assessing alternatives based on \( u_{MAU}(s) \))**

\[ u_{\text{max}} := 0, \quad a_{\text{max}} := \text{nothing} \]

for each \( a \)

\[ \text{for } (s = 1 \text{ to } #dp+#ap) \]

\[ \text{calculate } u_{\text{MAU}}(s) \]

\[ \text{if } (u_{\text{MAU}}(s) \leq u_{\text{max}}) \text{ skip } a \]

\[ \text{else calculate } u(s) \]

\[ \text{if } (u(#dp+#ap) > u_{\text{max}}) \]

\[ u_{\text{max}} := u(#dp+#ap) \]

\[ a_{\text{max}} := a \]

Winner of DDM is \( a_{\text{max}} \).

---

**Figure 4.6:** MAU algorithm.
For our investigations concerning the runtime performance tuning of DDM, let us consider the average case runtime performance as described in section 4.1. Let us further assume that \( t_{DDM}^q(n) \) is this runtime needed for one DDM process when the input data is of length \( n \). Let \( q_n(x) \) be a family of probability distributions for input data \( x \) of length \( n \). Let \( t_{DDM}(x) \) be the runtime of the DDM algorithm with input data \( x \). According to [98], the average case runtime performance of DDM can then be calculated as denoted in (4.16).

\[
    t_{DDM}^q(n) = \sum_{x|\|x|=n} q_n(x) \cdot t_{DDM}(x).
\]

Throughout our investigations, all interpreted values \( iv(p) \) of all involved decision and adaptable parameters are uniformly distributed within a range of \( 0 < iv(p) \leq 1 \). That means we only consider valid parameter values, i.e., values that do not cause DDM to regard a parameter to be irrelevant or an alternative to be unsuitable (see definition of the usefulness scale \( S_N \) in section 3.3). Consequently, \( q_n(x) \) is a uniform probability distribution for all \( n \) and all \( x \). Further work could address questions like which probability distributions are representative in which scenarios, and how does the probability distribution applied influence the average runtime gain (see section 6.3).

### 4.4.2 An example of DDM applying MAU

In this section, an example is presented to illustrate the usefulness calculation based on the MAU mechanism. Passing through this example helps getting a deep understanding of the operation of DDM and reveals clearly why, where, and under which conditions runtime savings can be achieved.

Let us assume a music video scenario similar to the one described in section 1.1.2. A user, Jane is looking for the Beatles song “Norwegian Wood”. Three alternatives \( (a_1, a_2, a_3) \) have been discovered in the internet, which are assessed by a DDM system applying the MAU mechanism. Jane considers four decision parameters as relevant for the concerned content retrieval decision making process:
An algorithmic approach to service and content deployment decision making

- $p_1$ is the price in Euros,
- $p_2$ is the sampling bit rate in kBit/s,
- $p_3$ is the retrieval success rate of the provider in percent (i.e., the percentage of how many of Jane’s former retrieval processes from the concerned provider succeeded), and
- $p_4$ is the reputation a provider earned by actually providing to Jane a service or content under the denoted conditions.

Price and sampling bit rate are decision parameters. Success rate and reputation are adaptable parameters instead. After discovering $a_1$, $a_2$, and $a_3$ as input for DDM, and selecting the parameters $p_1$ to $p_4$ in step 1 of the DDM algorithm, the usefulness for each alternative has to be calculated in step 2. In our example, all interpreted values of any parameters are chosen in a way that supports understanding the algorithm as good as possible. Details of parameter values themselves, or interpreters applied are omitted for reasons of conciseness. The usefulness calculation starts with $u_{\text{max}}=0$ and $a_{\text{max}}=$nothing. Alternative $a_1$ can be assessed completely because there is no concurring alternative $a_{\text{max}}$ with higher usefulness so far. The according usefulness $u_{a_1}$ results from accumulating the interpreted values of all four parameters in the following way:

$$u_{a_1} = \frac{0.5 + 0.9 + 0.8 + 0.8}{4} = 0.75.$$  

Table 4.3 illustrates these first steps of the usefulness calculation until the state where the processing of $a_1$ finished.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$u_{p_1}$</th>
<th>$u_{p_2}$</th>
<th>$u_{p_3}$</th>
<th>$u_{p_4}$</th>
<th>$u_a$</th>
<th>$u_{\text{max}}$</th>
<th>$a_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>0.5</td>
<td>0.9</td>
<td>0.8</td>
<td>0.8</td>
<td>0.75</td>
<td>0.75</td>
<td>$a_1$</td>
</tr>
</tbody>
</table>

Table 4.3: Calculation of $u_{a_1}$ applying MAU.

The usefulness calculation continues with assessing $a_2$ parameter-wise. Before computing the interpreted value of a parameter, the maximum attainable usefulness of the concerned alternative $u_{\text{MAU}}(s)$ is evaluated and
Algorithmic complexity analysis

compared to the current maximum usefulness $a_{\text{max}}$. Table 4.4 shows each step of this calculation for assessing $a_2$.

### Table 4.4: Calculating $u_{\text{MAU}}$ of $a_2$.

<table>
<thead>
<tr>
<th>$u_{p1}$=0.1</th>
<th>$u_{p2}$=0.3</th>
<th>$u_{p3}$=0.7</th>
<th>$u_{p4}$=0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{\text{MAU}}<em>{-a_2} = \frac{1+1+1+1}{4} = 1 &gt; u</em>{\text{max}}=0.75$? Of course, continue!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_{\text{MAU}}<em>{-a_2} = \frac{0.1+1+1+1}{4} = 0.77 &gt; u</em>{\text{max}}=0.75$? Yes, continue!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_{\text{MAU}}<em>{-a_2} = \frac{0.1+0.3+1+1}{4} = 0.6 &gt; u</em>{\text{max}}=0.75$? No, skip $a_2$!</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In step $s=1$, it is obvious that the alternative $a_2$ has a chance to win if the maximum attainable usefulness, i.e., 1 is assumed for each parameter. After step $s=2$, beating the current maximum usefulness of $u_{\text{max}}=0.75$ is still possible. However, after assuming MAU in step $s=3$ for alternative $a_2$, $u_{\text{max}}$ cannot be reached anymore, even if the remaining usefulness values for $p_3$ and $p_4$ would be 1. This discovery causes the DDM algorithm to skip $a_2$, and so, save the computational resources that would have been necessary for interpreting $p_3$ and $p_4$. Table 4.5 summarises the usefulness calculation of DDM until the state where the processing of $a_2$ finished.

### Table 4.5: Calculation of $u_{a_2}$ applying MAU.

<table>
<thead>
<tr>
<th>$a_1$</th>
<th>$u_{a_1}$</th>
<th>$u_{\text{max}}$</th>
<th>$a_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_2$</td>
<td>0.75</td>
<td>0.75</td>
<td>$a_1$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
As for $a_2$, the usefulness of $a_3$ is calculated parameter-wise. Again, the maximum attainable usefulness is evaluated and compared to $u_{\text{max}}$. Table 4.6 shows each single step.

<table>
<thead>
<tr>
<th>$u_{p_1}$=0.5</th>
<th>$u_{p_2}$=0.6</th>
<th>$u_{p_3}$=0.7</th>
<th>$u_{p_4}$=0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{\text{MAU}<em>{-a_3}} = \frac{1+1+1+1}{4} = 1 &gt; u</em>{\text{max}} = 0.75$? Of course, continue!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_{\text{MAU}<em>{-a_3}} = \frac{0.5+1+1+1}{4} = 0.875 &gt; u</em>{\text{max}} = 0.75$? Yes, continue!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_{\text{MAU}<em>{-a_3}} = \frac{0.5+0.6+1+1}{4} = 0.775 &gt; u</em>{\text{max}} = 0.75$? Yes, continue!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_{\text{MAU}<em>{-a_3}} = \frac{0.5+0.6+0.7+1}{4} = 0.7 &gt; u</em>{\text{max}} = 0.75$? No, skip $a_3$!</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6: Calculating $u_{\text{MAU}}$ of $a_3$.

Similar to $a_2$, one interpretation step was saved when assessing $a_3$, because in step $s=4$ it turned out that $a_3$ had no chance to attain a usefulness more the $u_{\text{max}}$. Consequently, $a_1$ won the DDM process with a usefulness of $u_{a_1}=0.75$ on the scale $S_N$. Table 4.7 shows the usefulness calculation of DDM after the processing of all alternatives finished.

<table>
<thead>
<tr>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$u_{p_1}$</th>
<th>$u_{p_2}$</th>
<th>$u_{p_3}$</th>
<th>$u_{p_4}$</th>
<th>$u_a$</th>
<th>$u_{\text{max}}$</th>
<th>$a_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.9</td>
<td>0.8</td>
<td>0.8</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>$a_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>0.3</td>
<td>saved</td>
<td>saved</td>
<td>-</td>
<td>0.75</td>
<td>0.75</td>
<td>$a_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
<td>saved</td>
<td>-</td>
<td>0.75</td>
<td>0.75</td>
<td>$a_1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7: Calculation of $u_{a3}$ applying MAU.
Summarising the presented example, the DDM algorithm saved 3 out of 12 interpretation steps which corresponds to 25% of all interpretation steps necessary for the regarded DDM process. The extent of a saving, of course, depends on the input data. It is possible that no step can be saved at all. However, assuming a uniform probability distribution for input values, the average case runtime performance clearly improves. This fact is shown with the help of an experimental evaluation in section 4.4.3.

4.4.3 Evaluation of runtime savings when using MAU

Following the example of applying MAU given in section 4.4.2, we can expect runtime savings for average case input data. However, we demand that this savings do not reduce the quality of the deployment or retrieval decision made. Accordingly, we state the following assertions.

Assertion 4.2

*When applying the mechanism of maximum attainable usefulness (MAU) to our DDM approach, the alternative with maximum usefulness is selected.*

Assertion 4.3

*Applying the mechanism of MAU to our DDM approach can save computational resources.*

The fact that Assertion 4.2 holds follows directly from the design of the MAU algorithm (see section 4.4.1). We only skip an alternative $a$, if there is no chance that the usefulness of $a$ exceeds the maximum usefulness that has been calculated so far in the concerned DDM process. All other parts of the DDM algorithm remain unchanged. To this end, the alternative $a_{max}$ exposing maximum usefulness will never be skipped, and so, will always be selected as the winner of DDM.

To find out more about the achievable performance improvement, and the factors influencing the yield of saved steps, we designed and carried out an experiment. As DDM itself, the behaviour of MAU depends on the relevant variables we determined (see Table 4.2). In our experiment, we do not apply any adaptable parameters, but decision parameters only. This is due to the fact that the MAU algorithm does not differentiate between
adaptable and decision parameters. Furthermore, we do not consider the number and kind of service parameters. Within the DDM algorithm, the consideration of MAU comes after matching decision and service parameters. So we can rely on the fact that for each decision parameter DDM utilises at the time when the MAU mechanism is computed, a corresponding service parameter is available. Consequently, we can consider the MAU mechanism as a function of the variables \( #a \) (number of alternatives to be assessed), and \( #dp \) (number of decision parameters that are involved).

In a first setting of our experiment to evaluate MAU we involve up to 100 alternatives and up to 100 decision parameters. All corresponding service parameter values are initialised randomly, which means that we use a generator for pseudo-random real numbers. This leads to equally distributed interpreted values \( iv \) within a range of \( 0 < iv \leq 1 \). This setting was simulated 100 times for each combination of \( #a \) and \( #dp \). Figure 4.7 and Figure 4.8 depict from two different perspectives the experimental results as percentages in a discrete space spanned by the variables \( #a \) and \( #dp \). The percentage \( sn \) denotes how many percent of the computational steps of calculating an interpreted value of any decision parameter involved in the regarded DDM process were necessary when applying MAU. The percentage \((100-sn)\) can be seen as a measure for the runtime saved in step 2 – calculating the usefulness of each alternative. Equation (4.17) is derived from (4.8) and expresses the influence of \( sn \) on the overall runtime of DDM.

\[
c_{\text{DDM, MAU}} \leq #a \cdot #a \cdot #sp_a \cdot #dp + #a \cdot #sp_a \cdot #cp + #ap + c_{\text{MAU}}(100-sn)\]

\( sn \): percent of value interpretations necessary in step 2 of DDM, \( 0\% \leq sn < 100\%\)

We call the percentage \((100-sn)=sav\), the percentage saved in terms of value interpretations. In Figure 4.7 it can be observed that the utilisation of MAU led to savings up to \( sav_{\text{max}}=45.02\% \). Furthermore, Figure 4.8 shows that \( sav \) increases with the number of alternatives \( #a \) that are assessed. And it decreases with the number of parameters involved \( #dp \), once a minimum of \( #dp \approx 4 \) is reached. The reason for this effect is that if more alternatives are assessed, the probability of finding an alternative with a high usefulness \( u_{\text{high}} \) increases. If \( u_{\text{high}} \) increases, the probability of skipping another
alternative with a usefulness value lower than $u_{\text{high}}$ increases as well. Moreover, we can say that if we utilise a smaller number of decision parameters, it is more likely that one extremely low interpreted value of a decision parameter leads to skipping the concerned alternative. If more decision parameters are considered, an extremely low interpreted value can be balanced easily by others. The experimental settings we investigated led to an average value of $s=85.36\%$. That means, we achieved an average saving in terms of value interpretation steps of $sav=14.64\%$. From the experimental observations we can conclude that applying the mechanism of MAU to our DDM approach can save computational resources. Consequently, Assertion 4.3 holds within the context of the performed investigation.

![Figure 4.7: Percentage of value interpretation steps depending on $\#a$ and $\#dp$, perspective 1.](image-url)
An algorithmic approach to service and content deployment decision making

Figure 4.8: Percentage of value interpretation steps depending on \( \#a \) and \( \#dp \), perspective 2.

Looking at Figure 4.8, we can observe that most runtime is saved when \( \#dp < 10 \) and \( \#a \) continues growing. To gain deeper insight into the relation between input data configuration and increase in performance gain we repeated the experiment described above applying a second set of settings. We simulated 1 up to 1000 alternatives and not more than 7 decision parameters, where each combination of \( \#a \) and \( \#dp \) was run 100 times. All other settings remained unchanged. Figure 4.9 depicts the results we obtained, especially emphasising the influence of \( \#dp \). It can be seen that the lowest value for the percentage of interpretation steps \( sn \) is obtained when exactly 4 decision parameters are applied. This fact holds for any number of alternatives \( \#a \). The minimum value \( sn_{\text{min}} \) we achieved in our experiment is \( sn_{\text{min}} = 40.09\% \). This corresponds to a saving of interpretation steps of \( sav_{\text{max}} = 59.91\% \). Figure 4.10 shows the same experimental results focusing on how \( \#a \) affects the runtime performance. It can clearly be observed that \( sn \) decreases when \( \#a \) increases. In detail, we see that \( sn \) asymptotically approaches a value of 40%.
Figure 4.9: Percentage of value interpretation steps, emphasising the influence of $\# dp$. 

Figure 4.10: Percentage of value interpretation steps, emphasising the influence of $\# a$. 
Taking as a basis a uniform probability distribution for interpreted decision parameter values, we can conclude that in the average case, and depending on the input data size \( \#a \) and \( \#dp \), up to 60\% value interpretation steps will be saved in step 2 of the DDM algorithm. A task for future work is to deduce analytically a mathematical relation for the data we obtained in our experiments (see section 6.3). Given that such a relation exists (which is indicated by our experimental results), the application of the MAU mechanism to other similar algorithmic problems can lead to considerable average case runtime savings. Affected may be the class of all algorithms that subsequently evaluate a set of variables (like \( \#dp \) in DDM) for a variety of entities (like \( \#a \) in DDM). These could be complex operations on, for instance, matrices or similar structures of higher dimensionality. A deeper investigation of conditions and consequences can clarify the potential of MAU and possibly related mechanisms.

Concluding on the presented investigations of applying the mechanism of maximum attainable usefulness to DDM, we can state the following findings. After explaining the principles of MAU, we illustrated with an example in which way the average case runtime performance of DDM can be improved. We carried out an experiment with different settings which confirmed the effects on performance tuning. We argued that running DDM without MAU always leads to selecting the alternative with highest usefulness value. Moreover, DDM calculates a usefulness value for each involved alternative. Using a sorting algorithm, we obtain a complete ranking of all alternatives according to their usefulness. Applying the MAU mechanism, DDM still always selects the alternative with highest usefulness (Assertion 4.2 holds). However, because DDM skips alternatives that cannot exceed a temporary maximum usefulness value, we cannot longer create a complete ranking for all alternatives according to their usefulness. For DDM, this is irrelevant, but if the DDM algorithm is adapted to further application fields, this fact has to be considered. Utilising MAU can save up to 60\% of value interpretation steps in step 2 of the DDM algorithm. To this end, the average case runtime of DDM applying MAU is reduced accordingly (Assertion 4.3 holds). The extent of saved steps depends on the number of regarded decision parameters and the number of assessed alternatives.
In section 4 we analysed the algorithmic complexity of our DDM approach. After defining fundamental terms and notations, we determined a set of variables which matter for the DDM algorithm’s complexity. These variables comprise the number of assessed alternatives $\#a$, the number of applied decision parameters $\#dp$ as well as adaptable parameters $\#ap$, and the number of service parameters that are exposed for each alternative $\#spa$. Based on that, we deduced analytically how much computational effort needs to be spent in each step of the DDM algorithm. We showed that the costs of step 5, adapting the involved adaptable parameters, depend on the user implementation of according interpreters. In case this implementation consumes reasonable costs, the whole DDM algorithm works efficiently, which means, it has polynomial worst case runtime complexity as we demanded in section 1.1.3. Subsequent to the complexity analysis, we elaborated on a variety of decision parameter selection methods that can be used to influence the computational costs of DDM. Multiple experimental evaluations showed clearly that applying such a parameter selection method is a trade-off between spending computational resources and getting the most useful deployment or retrieval decision. Additionally, we presented the mechanism of maximum attainable usefulness (MAU) that, if applied, leads to savings of value interpretation steps of up to 60%. This means, applying MAU improves the average case runtime performance of DDM. At the same time, we deduced that the quality of deployment or retrieval decisions is not altered by using MAU. DDM will always select the alternative with maximum usefulness. The relation between the number of involved decision parameters, the number of alternatives assessed, and the number of value interpretation steps saved, was evaluated experimentally, which confirmed all assertions we stated accordingly.
5  Adaptability

In the preceding sections of the thesis at hand we presented an
approach to come to content retrieval and service deployment decisions in a
way that fits best to user preferences and needs. Moreover, we demanded
that our approach works in highly complex, changing, and partially
autonomic computer network environments. We analysed the algorithmic
complexity of our approach and derived several properties analytically or
experimentally. In this section, we explain how to align our DDM approach
with what a user prefers and how the behaviour of DDM can be influenced
accordingly. We moreover explain a facility that enables DDM to react to,
for example, environmental changes without the need of explicit user
intervention. To this end we incorporated a set of facilities into our DDM
approach which can be categorised as user- and self-adaptation
mechanisms. Both ways of adapting DDM are presented, explained, and
evaluated in the following sections (see also [103] and [104]).

5.1  User-adaptability

With user-adaptability we summarise all features that enable a DDM
user to influence the behaviour of dedicated decision making processes. The
provision of user adaptability features enables a user to configure our DDM
approach according to individual prerequisites and needs, as well as
preferences and priorities concerning factors that are important within the
scope of a decision making process. This implies that proper adaptation
done by a user is a basic requirement for meaningful content retrieval and
service deployment decisions. As demanded in section 1.1.3, we want our
DDM approach to work as much as possible independent of the need for
user intervention. To this end, user-adaptation is only possible before or
after, but not in the course of one DDM process. This means, configurations
and adjustments done are valid for at least one decision. However, if no
need is given to change any aspects of DDM, multiple decision making
processes can be run without any user-adaptation in between. Our DDM
approach provides three facilities for being adapted by users, firstly,
selecting representative parameters, secondly, applying individual
interpreters, and thirdly, adjusting parameter weights. Each of these
facilities is explained within the following sections.
5.1.1 Selecting parameters

Description, evaluation, and selection of deployment or retrieval alternatives are based on the parameter concept defined in section 3.2.3. Alternatives themselves and the conditions under which they can be obtained are represented in terms of service parameters. Aspects that matter for a DDM user are expressed utilising decision and adaptable parameters, where the former evaluate service parameter values, and the latter assess collected experiences. Both, decision and adaptable parameters determine the usefulness of the aspect they represent based on an interpreter that maps a service parameter value or experiences to the usefulness scale $S_N$ (see section 3.3). It has to be considered that a decision parameter can only be regarded if a matching service parameter is exposed by the alternative assessed. Otherwise, it is either ignored or leads to complete rejection of the concerned alternative. Which of the two cases applies depends on how the interpreter of the decision parameter is implemented.

With regard to affecting the algorithmic runtime of DDM (see section 4), any number and kind of decision and adaptable parameters can be utilised for a decision making process. Moreover, users are encouraged to design, implement and incorporate own parameters to individually represent any aspect they consider to be relevant in each domain they apply DDM to. How parameter selection affects the computational resource consumption as well as the usefulness of deployment or retrieval decisions can be understood by studying section 4.3.

Besides representing aspects of concern, selecting an appropriate set of parameters can be used to adapt to different content or service description formats. As shown in sections 2.1.8 and 3.2.3, a variety of possibilities to describe content or services exist now or may exist in the future, depending on application domains and underlying technologies. We designed our DDM approach to be as much as possible independent from any dedicated description specification, and we achieve this by enabling the incorporation of arbitrary, user implemented decision and adaptable parameters. The only conditions those parameters have to meet is that they implement an interpreter that maps the parameter’s value to the usefulness scale $S_N$ in the way defined in section 3.3. As the algorithmic complexity of user specific adaptable parameters influences the overall runtime performance of DDM, it is recommended to implement those parameters in line with the user’s demands on the performance and efficiency (see section 4.2).
5.1.2 Applying individual interpreters

For the fact that adaptable and decision parameters can represent whichever aspect of concern for any user applying DDM, a parameter’s value range is not restricted. It rather can comprise arbitrary information that can be symbolic (like a text), sub-symbolic (like a set of numbers representing raw sensor data), it can be mathematically denoted (like a system of equations), a picture (of a desired consumer good), and so on. To be able to process this variety of information, it is mapped to the usefulness scale $S_N$. This is put into practice by an interpreter. Each decision and adaptable parameter can apply exactly one interpreter at a time. However, one and the same aspect, i.e., parameter might have different interpretations which can depend on the user applying the parameter, the domain a DDM process is concerned with, and the situation in which a user currently considers the parameter to be relevant. With regard to this fact, we designed our DDM approach in a way that, as parameters themselves, also their interpreters are exchangeable. That means, a user can design, implement, and plug in individual interpreters to any parameter he or she takes into account for a DDM process.

To illustrate this facility, consider for example the decision parameter sampling bit rate of a music file as in the scenario described in section 1.1.2. Jane, who likes watching music videos very much and is well equipped with storage space on her laptop, might prefer high quality music with a sampling bit rate of 192 kBit/s. A lower or a higher bit rate causes the usefulness of an alternative concerning this decision parameter to decline. John instead, rarely listens to music and is very limited in storage space. He might only accept music files with a bit rate of exactly 64 kBit/s. Jane and John can easily adapt our DDM approach to their own preferences on the bit rate of music files by applying individual interpreters accordingly.

To demonstrate the effect, and to get an impression of the applicability of our approach, we carried out a real world experiment. We based this experiment on a video download scenario similar to the one described in section 1.1.2. Two existing providers of free video content in the internet were queried for different music videos. Communication with the content providers took place by utilising application programming interfaces they offered. Dedicated decision parameters have been developed to make retrieval decisions based on metadata the providers use to describe their content. These parameters were DateOfPublication, which denotes when a video was published, and Duration, which corresponds to the length of a
An algorithmic approach to service and content deployment decision making

video in seconds. Moreover, we applied an adaptable parameter ProviderReputation. For reasons of conciseness, we kept the value of this last parameter constant throughout the whole experiment. Ten rounds, i.e., repetitions were performed, where a different song was requested in each round. As input for DDM we took into account the first query replies of each provider.

In a first decision making process we plugged into the decision parameter DateOfPublication an interpreter that prefers newer videos instead of older ones. That means newer alternatives have higher interpreted values, i.e., partial usefulness for the concerned parameter. Figure 5.1 depicts the results of this experiment. On the y-axis, the overall usefulness of an alternative is mapped to the scale $S_N$. On the x-axis, the service parameter values for DateOfPublication are denoted from older to newer dates on a discontinuous scale. Accordingly, each alternative that won a round is represented as stack of bars, where a white bar corresponds to the partial usefulness of ProviderReputation, a gray bar corresponds to the partial usefulness of Duration, and a black bar corresponds to the partial usefulness of DateOfPublication. Superposing all partial usefulness values delivers the overall usefulness of an alternative. What can be observed in Figure 5.1 is the fact that a newer date of publication causes a higher partial usefulness of the corresponding decision parameter. We repeated the experiment with exactly the same settings except the interpreter of DateOfPublication. For this second evaluation, we preferred older videos, meaning that an older date of publication causes a higher partial usefulness value of the corresponding decision parameter. Figure 5.2 depicts the results accordingly and confirms exactly this modified interpretation. Comparing Figure 5.1 and Figure 5.2 shows two facts. Firstly, the application of different interpreters leads to dissimilar usefulness values for one and the same aspect of concern, i.e., decision parameter. Secondly, we can perceive that the values of DateOfPublication, which the selected alternatives expose, differ. For example, the values “2007-12-08”, “2008-06-07”, etc. only appear in the first run of the experiment. Given that both DDM processes were performed, except the interpreter of DateOfPublication, with exactly the same settings, this implies that different alternatives won the decision making processes. Consequently, we can conclude that applying individual interpreters can lead to different results of a DDM process.
Figure 5.1: Requesting ten music videos from the internet, newer content is preferred.

Figure 5.2: Requesting ten music videos from the internet, older content is preferred.
The findings of the performed experiment confirm that the described adaptation facility can be used to align DDM with individual preferences of users. Moreover, we experienced that an application of our approach is possible and does work within a real word domain. Especially with regard to media retrieval, we already can find providers exposing accessible content. Further than that, a variety of metadata coming along with provided content can be utilised as service parameters based on which multiple alternatives can be assessed by our DDM approach.

5.1.3 Adjusting parameter weights

The usefulness calculation of the DDM algorithm is based on a weighted sum of the interpreted values of all involved adaptable and decision parameters, divided by their overall number (see (3.6)). In this calculation, each parameter has its own weight. Changing a weight means to decrease or increase the influence the concerned parameter has on the overall usefulness value of the associated alternative. Consequently, a parameter weight expresses how important a parameter is, compared to others. Adjusting weights means to influence the importance of parameters. If a user considers for instance the date of publication of music video content more important than other parameters but does not want to completely abstain from considering video duration, provider reputation, and so on, he or she can increase the influence of the date of publication by raising its weight above the weights of other parameters.

To illustrate the effect of adjusting parameter weight as a facility to adapt DDM, we carried out an experiment similar to the one described in section 5.1.2. We again queried two real world providers offering free music video content in the internet. Once more, we applied the decision parameters $\text{DateOfPublication}$ and $\text{Duration}$, as well as the adaptable parameter $\text{ProviderReputation}$, where the value of $\text{ProviderReputation}$ remained unchanged throughout all repetitions of the whole experiment. The experiment was repeated ten times with exactly the same query for one desired song. We set the weights of the applied parameters to the following values:

$$w_{\text{Duration}} = 0.5$$
$$w_{\text{ProviderReputation}} = 0.5$$
$$w_{\text{DateOfPublication}} = \text{varying from 0.1 up to 1, increasing by 0.1 per round}.$$
The experimental results are depicted in Figure 5.3. The product of the interpreted value of the applied parameters and the associated weight is mapped to the y-axis. The x-axis shows the varying weight of the parameter DateOfPublication. All partial usefulness values of each parameter of the alternatives selected as winner of DDM in each round are shown. It can be observed that an increasing weight $w_{\text{DateOfPublication}}$ causes the influence of the decision parameter DateOfPublication on the DDM decision to increase as well. In round eight when a value of $w_{\text{DateOfPublication}} = 0.8$ is reached, we can notice that the increasing weight caused a change in the content retrieval decision. Even though the partial usefulness values of the parameter Duration is much lower compared to the alternative selected in the preceding rounds, the partial usefulness of DateOfPublication had sufficient influence to cause the decision made. These results show that adjusting adaptable and decision parameter weights influences retrieval and deployment decisions made by DDM, and can for that fact be used to adapt DDM to user priorities.

Section 5.1 illustrated three facilities a user has to adapt our DDM approach to individual preferences and needs. In all mentioned cases, adaptation has to be performed manually by the user. In section 5.2 we
elaborate on how DDM can adapt itself to changes without the need for user intervention.

5.2 Self-adaptability

In section we characterised 3.2.1 the nature of information our DDM approach has to deal with if we envisage application in a complex computing network domain. One characteristic for concerned information is its unreliability. Data DDM relies on might be altered in the following three ways:

1. Information can be incomplete or missing. In terms of DDM, this may for example be the case when properties of alternatives are denoted in an insufficient manner. Certain service parameters being available for only a few of all alternatives that are regarded is such an incompleteness of information.

2. Information can be wrong because it was modified, without purpose by any unforeseen effects. A network fault or an error affecting an involved entity can cause such an accidental modification. Moreover, information can unintentionally be incorrect because it simply is outdated. This effect is likely to occur in fast changing complex networks that process huge quantities of data by a large amount of entities.

3. Information can be wrong because an entity manipulated it purposely. With regard to DDM, a provider may have the interest to increase its sales figures and it is trying to achieve this by promoting untrue service parameter values for services or content it offers.

In the scope of DDM, a user can counter information incompleteness by selecting decision and adaptable parameters properly. If wanted, common parameters (see section 4.3.2) can be used to avoid missing service parameters for some of the assessed alternatives. Of course, this facility can only answer the problem of incomplete information to a limited extent. If few or no common parameters are available at all, the issue remains unsolved. Manipulating information with or without purpose can be handled, to a certain extent, by applying adaptable parameters. These represent experiences that have been gathered before. Each instance of a DDM system collects only information about deployment or retrieval
processes that this instance was involved in itself. Sharing experiences among different instances of DDM is possible by exchanging the values of adaptable parameters. However, questions of experience distribution or community learning have not yet been considered within the scope of our research work. Further investigations may discover according properties, possibilities, and limitations (see section 6.3).

To investigate the potential of applying experience gathering for self-adaptation purposes, we developed two examples which are described in the following sections. Firstly, we consider an adaptable parameter that represents the reputation of service and content providers. Secondly, an adaptable parameter for observing the success rate of deployment and retrieval processes is regarded.

### 5.2.1 Provider reputation

Decisions in the scope of DDM rely, to a large extent, on service parameter values that are denoted by service and content providers. To be able to estimate the trustworthiness of such information a user can evaluate his or her experiences made with providers during former DDM processes. We developed an adaptable parameter ProviderReputation that assigns a reputation value to providers, a DDM system was in contact with. For this purpose, ProviderReputation constructs a map of all providers that won a DDM process, associated to a floating point value $r_p$ with $0 < r_p \leq 1$, that represents a provider’s reputation. Learning takes place by comparing service parameter values that were denoted by providers, with all corresponding measured values that were observed during service deployment or content retrieval. For this purpose, two usefulness values are calculated for the winning provider. The first, $u_s$ is based on stated service parameter values. The second, $u_m$ relies on measured values. Both usefulness values are compared to each other. In the case that $u_m$ is greater or equal to $u_s$, the reputation value of the affected provider is increased. The degree of increase is determined by a learning rate $r_l$ with $0 < r_l \leq 1$, which can be specified by the user. If $u_m$ is slightly less than $u_s$, the reputation $r_p$ remains unchanged. How much deviation is accepted is determined by the user in terms of specifying a tolerance $t$ with $0 \leq t < 1$. In case, $u_m$ is smaller than $u_s$ by more than the tolerance $t$, the reputation $r_p$ of the regarded provider is decreased by $r_l$. Let $r_p(s-1)$ be the reputation value of the
An algorithmic approach to service and content deployment decision making

Adaptable parameter *ProviderReputation* concerning a provider *p* directly before *p* wins a DDM process. The value *rₚ*(s) represents the reputation value of *p* after adapting *ProviderReputation* in step 5 of the DDM algorithm (see Figure 3.2) as denoted in (5.1).

\[
\begin{align*}
    rₚ(s) &= \begin{cases} 
    rₚ(s-1) + rᵢ, & \text{if } uₘ \geq uₛ \\
    rₚ(s-1), & \text{if } uₘ \geq uₛ - t \text{ and } uₘ < uₛ \\
    rₚ(s-1) - rᵢ, & \text{if } uₘ < uₛ - t
    \end{cases} 
\end{align*}
\]  \hspace{1cm} (5.1)

When the adaptable parameter *ProviderReputation* is evaluated during a DDM process, its reputation map can comprise as many entries as providers won a decision making process calculated by the concerned DDM system. To this end, large amounts of computational resources have to be spent if *ProviderReputation* embraces many providers. To avoid inappropriately much computational expense, the maximum number of providers stored in the reputation map might be limited. Even though we have not implemented any size restriction mechanisms for *ProviderReputation*, various strategies are possible. For example, the most rarely requested, or the oldest providers can be removed from the reputation map if a size limit is reached. *ProviderReputation* is illustrated in terms of a class diagram in Figure 5.4.

![Class diagram of ProviderReputation](image)

**Figure 5.4:** Class diagram of *ProviderReputation*. 

- 122 -
5.2.2 Evaluation of provider reputation

To evaluate the behaviour of the adaptable parameter ProviderReputation within DDM processes we simulated a music download scenario similar to the one presented in section 1.1.2. The results obtained show how reputation values of regarded providers evolve based on the experiences gained among multiple DDM processes.

In a first simulation we investigated the development of ProviderReputation when three content providers offer a desired music file under well defined, similar conditions. We applied the two decision parameters sampling bit rate in kBit/s and price in Euros. All values of all corresponding bit rate service parameters were initialised equally with a value of 192 kBit/s. All values of price of Provider1 and Provider3 were set to 0.99 Euro. In our simulation we defined that Provider2 always exposes a price of 0.09 Euros but charges 0.99 Euros after content retrieval. Such behaviour can correspond either to cheating or to unintentionally denoting wrong service parameter values. Within the context of our simulation, it does not matter which of the two cases caused Provider2 to expose an incorrect price. We rather want to find out if the adaptable parameter ProviderReputation, which we applied in addition to bit rate and price, is capable of handling this situation. All reputation values were initialised with a value of 0.5. Moreover, they had a learning rate of \( r_l = 0.1 \) and a tolerance of \( t = 0.01 \). Figure 5.5 shows the results of the described simulation over a period of ten rounds. It can be observed that Provider2 was selected as winner in rounds 1 and 2. This is due to the fact that the overall usefulness of the alternative it offered exceeded all others because of a lower price. However, the deception of Provider2 was recognised after each round which caused its reputation value to decrease by the learning rate \( r_l = 0.1 \). In the third round, the reputation value of Provider2 was too low so that, despite a lower denoted price, Provider1 was selected as winner instead. Because of the fact that Provider1 denoted price and bit rate values absolutely correct, its reputation was increased by \( r_l = 0.1 \). Consequently, the renowned Provider1 was selected again in all following rounds.
We conducted a second simulation in which we investigated how \textit{ProviderReputation} affects DDM when multiple providers have different service parameter values. We applied again the adaptable parameter \textit{ProviderReputation} as well as the decision parameters bit rate and price. For price and bit rate we applied the interpreters described in section 3.4.1 with the difference that bit rates $320 < b \leq 512$ kBit/s have interpreted values of $iv = 2^{-1074}$, i.e., slightly greater than 0. \textit{ProviderReputation} was initialised with a value of 0.5, a learning rate of $r_l = 0.1$, and a tolerance of $t = 0.01$. All bit rate values were set randomly, using a pseudo random number creator, to 64, 128, 192, 256, 320, or 512 kBit/s. All prices were fixed to 0.99 Euros except for \textit{Provider1} who always denoted a price of 0.09 but charged 0.99 Euros after winning a DDM process. In this simulation, we added five providers at all, each of them offering a desired music file. Figure 5.6 depicts the results obtained by performing DDM over 20 rounds. The lines connecting a provider’s interpreted reputation values emphasise the changes over all rounds. The diagram shows that only the reputation of \textit{Provider1}, who denoted incorrect prices, decreased. In contrast, the reputation of all other providers increased or remained stable. A further observation is the fact that, despite an already low reputation value $r_{p1} = 0.3$ of \textit{Provider1}, this provider was selected as winner by DDM again in round 18. The reason for

![Figure 5.5: Analysis of ProviderReputation, Provider2 denotes an incorrect price.](image)
this effect is that the overall usefulness of all service and adaptable parameter values of Provider1 exceeded the usefulness of all others. As a consequence we can perceive that even if a provider has a bad reputation, it might get a chance to rehabilitate itself. However, in our simulation Provider1 continued denoting incorrect prices which led to further decrease of its reputation.

As results of the evaluation described in this section we can summarise that applying the adaptable parameter ProviderReputation affects deployment and retrieval decisions made by DDM. Describing alternatives correctly in terms of service parameters leads to increasing the reputation of the alternative’s provider. Denoting incorrect service parameter values leads to a reputation decrease. DDM prefers alternatives offered by providers with a high reputation value. Nevertheless, gathering reputation experiences is a slow learning mechanism. Only one learning step for one provider takes place after this provider won a DDM process. We can conclude that applying ProviderReputation as adaptable parameter enables DDM to react to recurrently incorrect notations of service parameter values. This self-adaptation facility is suitable when many DDM processes are calculated and a limited number of different service or content providers are involved. Using ProviderReputation is not a sufficient mechanism to completely avoid deception or incorrectness in denoted service parameter values. The investigation and development of more advanced reputation
mechanisms, like exchanging reputation values among multiple DDM systems or building reputation based on community experiences, can be part of further work (see section 6.3).

### 5.2.3 Success rate

As a second example besides ProviderReputation, we developed the adaptable parameter SuccessRate to investigate how self-adaptability of DDM can be used to address the problem of unreliable information. SuccessRate represents how many of all deployment or retrieval processes, a provider was involved in, were successful. To gather experiences made, SuccessRate builds a map which contains all providers that were selected as winner of a decision making process. Each provider \( p \) is associated with a success rate \( sr_p \) with \( 0 \leq sr_p \leq 1 \), and the number \( n_p \) of overall decision making processes \( p \) won. Learning takes place after each service deployment or content retrieval process. Information about the winning provider and about success or failure of the preceding deployment or retrieval is passed to SuccessRate. As a consequence, the success rate \( sr_p \) and the number of interactions \( n_p \) are adapted accordingly as denoted in (5.2). Let \( n(s-1) \) be the number of DDM processes provider \( p \) won directly before the current DDM process. Let \( n(s) \) be the number of DDM processes \( p \) won including the current DDM process, where \( 1 \leq s \).

\[
\begin{align*}
n(s) &= n(s-1) + 1 \quad \text{and} \\
\frac{sr_p(s-1) \cdot n(s-1) + 1}{n(s)} &\quad \text{if deployment / retrieval succeeded} \\
\frac{sr_p(s-1) \cdot n(s-1)}{n(s)} &\quad \text{if deployment / retrieval failed} \\
\end{align*}
\]

(5.2)

Similar to ProviderReputation, the size of the map containing providers and success rates affects the computational effort that is required to evaluate the adaptable parameter SuccessRate. For this fact, the number of providers stored in the success rate map might be limited. We have not established any limit for our implementation of SuccessRate. However, forgetting the least frequently used provider or the provider with the least number of DDM processes won are possible strategies to avoid too large
success rate maps. Figure 5.7 shows a class diagram of our implementation of \textit{SuccessRate}.

![Class diagram of SuccessRate](image)

**Figure 5.7:** Class diagram of \textit{SuccessRate}.

### 5.2.4 Evaluation of success rate

We evaluated the effect of applying the adaptable parameter \textit{SuccessRate} within our DDM approach by simulating again a music retrieval scenario. Besides \textit{SuccessRate}, we utilised the sampling bit rate and the price as decision parameters. In a first attempt, we set up three providers by initialising all prices with pseudo random values $v_p$, equally distributed within a range of $0 \leq v_p \leq 5$ Euros. All bit rates were set to pseudo random values out of the set $V_b = \{64, 128, 192, 256, 320, 512\} \text{ kBit/s}$. For each provider, we initialised \textit{SuccessRate} with $n_p(0) = 0$. DDM process won before, and $sr_p(0)=0$, which expresses that no DDM process was successful so far. The performed simulation was designed in a way that Provider1 succeeds in retrieving a desired music file after it won a DDM process with a probability of $\text{prob}_{p1} = 0.3$. Provider2 succeeds with $\text{prob}_{p2} = 0.6$ and Provider3 succeeds with $\text{prob}_{p3} = 0.9$. 
Figure 5.8 depicts the results we obtained after 30 rounds of DDM. The lines connecting \( \text{SuccessRate} \) values emphasise the development over time. It can be observed that after a provider won a DDM process for the first time, it gained an extreme value for \( \text{SuccessRate} \). This value is either 0 or 1, which means the provider never or always finished music retrieval successfully. After a small number of rounds, each provider achieves a success rate that corresponds to its predefined success probability \( \text{probp}_1 \), \( \text{probp}_2 \), or \( \text{probp}_3 \). These results confirm within the scope of the simulated scenario that the adaptable parameter \( \text{SuccessRate} \) appropriately represents how reliable a provider carries out deployment or retrieval processes. However, it has to be noted that \( \text{SuccessRate} \) values do not necessarily converge towards their predefined success probabilities. This is due to the fact that a low success rate, for example after the first deployment or retrieval process of a provider failed, might prevent DDM from selecting an associated provider often enough.

To overcome the effect of extreme starting success rates, we simulated the described scenario again using dissimilar initial values. All settings in this second simulation remain unchanged, except \( n_p(0) \) which we redefined to be \( n_p(0)=5 \), and \( sr_p(0) \) which we set to \( sr_p(0)=0.5 \). The results obtained are presented in Figure 5.9. The simulation results show that the success rate of each provider evolves towards the previously defined success
probability. From this fact we can conclude that, again, the adaptable parameter \textit{SuccessRate} reflects how reliable a provider carries out deployment or retrieval processes. We do not observe extremely high or low success rates after a provider was selected for the first time. Figure 5.9 rather indicates a success rate development based on small gradual steps. Because an increasing number $n_p$ of overall won DDM processes causes the success rate modification to asymptotically approach 0 (see (5.2)), defining a maximum value for $n_p$ might be appropriate. We have not applied such a restriction in our implementation of \textit{SuccessRate}. However, a user who wants to reuse and individualise this adaptable parameter can incorporate an according modification. Even though the initial settings of $n_p(0)=5$ and $sr_p(0)=0.5$ caused \textit{SuccessRate} not to take extreme values, an exact convergence towards the defined success probability cannot be granted.

![Figure 5.9](image)

\textbf{Figure 5.9: Analysis of SuccessRate, initialisation with $n_p(0)=5$, $sr_p(0)=0.5$, $prob_{p1}=0.3$, $prob_{p2}=0.6$, $prob_{p3}=0.9$.}

In section 5.2 we presented a facility for self-adaptation of DDM. We exemplified its properties, potential, and limitations with the help of two adaptable parameters, \textit{ProviderReputation} and \textit{SuccessRate}. The simulations carried out show that, utilising adaptable parameters, DDM has the capability to react to unreliable information. Involving \textit{ProviderReputation} and \textit{SuccessRate}, the DDM algorithm preferably selects alternatives which are offered by providers that describe conditions of the services or content they offer correctly. Moreover, alternatives are considered to be more useful
if the offering provider was experienced to carry out deployment or retrieval processes successfully. By showing the influence of self-adaptation on decision making processes, we fulfilled the demand of enabling DDM to react to unreliable information to a certain extent (see section 1.1.3). Nevertheless, we also showed limitations of applying adaptable parameters. In our implementation of ProviderReputation and SuccessRate, experiences are gained after each DDM process and only affect the provider that won. Developing additional adaptable parameters which, for example, observe trends in the notation or measurement of service parameter values might further improve self-adaptability of our DDM approach (see section 6.3).

5.3 Summary

After presenting the DDM approach to solve the deployment decision making problem stated in section 1.1.3, and subsequent to analysing the algorithmic complexity of this approach, we elaborated on adaptation mechanisms of DDM in section 5 of the work at hand. Two different categories of adaptation were described, first, adaptation done by users, and second, adaptation by DDM itself without explicit user intervention. The former kind, user-adaptation, can be achieved by selection appropriate sets of decision and adaptable parameters, by applying individual interpreters, and by adjusting parameter weights. The latter kind, self-adaptation, is put into practice by involving adaptable parameters like ProviderReputation or SuccessRate into DDM processes. Experiments based on existing internet video providers, as well as artificially designed simulations showed that user-adaptability can be utilised to align DDM with individual preferences and needs, depending on users, environments, and situations. Moreover, self-adaptability equips the DDM approach with the ability to react autonomically to unreliable information in changing environments as we demanded in section 1.1.3. Even though the problem of unreliable information is not completely solved by the facility of adaptable parameters, the potential of achieving more useful DDM results is confirmed.
6 Conclusion and outlook

In this section, we summarise the thesis. After reprising the content of each chapter, we discuss findings and contributions achieved. Finally, we provide an outlook on open issues and further work.

6.1 Summary

In this thesis we determined and investigated the deployment decision making (DDM) problem. In section 1, we stated that this problem comprises the task of deciding, in a meaningful and user-specific manner, how to select one out of many alternatives to deploy a desired service or retrieve wanted content. We showed the need for a solution by presenting two scenarios, firstly, being concerned with autonomic, context aware advertisement, and secondly, supporting a user in obtaining music video content. Subsequently, we listed a set of demands we impose on our approach to solving DDM. Next, we delimited our work from adjacent topics and explained the methodology applied. We thereafter derived the contributions of our work to theory and practice of computer science.

In section 2, we briefly summarised related work and activities. By considering related research topics, and by presenting two exemplary projects, we determined the position of DDM within a broad scientific scope. Afterwards, a list of periodically occurring challenges was compiled to show current trends and activities in the fields of autonomic and service oriented computing.

In section 3, we presented our approach to solving the DDM problem. We denoted the terminology used, elaborated on needed and available information, and defined a set of parameters we use to represent any information that is processed. Next, we explained in detail the DDM algorithm which is applied to compute deployment decisions, and a metric this algorithm utilises for assessing the usefulness of deployment alternatives. The function of the presented algorithm was illustrated with the help of three simulations of a music video scenario similar to what we described in section 1.1.2.

To judge the applicability and efficiency of the proposed algorithm, we performed an analysis of its algorithmic complexity and denoted the results in section 4. Firstly, we briefly explained underlying complexity
theory and clarified assumptions we made. Secondly, the algorithmic complexity of DDM was deduced step by step. Thirdly, we showed that selecting subsets of parameters for assessing alternatives influences the complexity as well as the quality of a calculated deployment decision. Fourthly, multiple parameter selection methods were presented and their impact was evaluated experimentally. And fifthly, we presented the principle of maximum attainable usefulness (MAU), illustrated it with a detailed example, and showed with the help of a simulation that applying MAU can improve the average case runtime of DDM by saving up to 60% of value interpretation steps in step 2 of the DDM algorithm.

Various facilities to adapt the behaviour of DDM were presented in section 5. We distinguished between user- and self-adaptation. The former, user-adaptation, can be achieved, firstly, by selecting appropriate sets of decision and adaptable parameters which represent aspects a user considers relevant for a deployment decision, secondly, by applying individual interpreters which express exactly, how useful an aspect is for a user, and thirdly, by adjusting parameter weights which leads to prioritising some parameters over others. The latter, self-adaptation, is put into practice by applying adaptable parameters to incorporate experiences into the DDM processes. Automatically adapting the reputation of service and content providers, and calculating a success rate for deployment processes were described to illustrate the potential and the limitations of using self-adaptation to address unreliability of information within the scope of DDM. The facilities for adaptation were evaluated partially based on simulations and partially based on real world experiments in which we incorporated video content providers available in the internet.

Finally, we summarise our work, and discuss findings as well as contributions in section 6, before we close this thesis with an outlook on open issues.

6.2 Discussion

This thesis contributes to the theory of computer science, especially autonomic and service oriented computing by providing definitions and detailed explanations of the deployment decision making problem, and a parameter concept for representing information needed for DDM. Of course, these definitions are relevant for the work we present because they build the basis for our approach to solving the DDM problem. Beyond this,
they might help illustrating the task of service selection seen from a user-centric point of view, and generalised to selecting content as well. The parameter concept deepens the understanding of which information is needed and which is available for a user interacting with service and content providers in computer networks. Others can use and adapt this parameter concept to structure and represent information in similar application domains for comparable tasks.

The usefulness scale we defined along with the mechanism of exchangeable parameter interpreters relies on a fundamental assumption we make: a user is capable of expressing exactly which aspect matters for him or her in which way. This assumption enables expressing any value of any parameter in terms of usefulness according to the metric we defined. Only if this assumption holds, we are able to compare independent aspects like price, colour, and fuel consumption of a car, because DDM does not rely on their values but on how useful a user considers each value. Designing parameters in a way that interpreters can completely be exchanged enables individual judgement of aspects depending on who judges them in which situation. However, with imposing the task to develop interpreters to the user, we ask he or she to spent considerable effort.

The fact that parameters are exchangeable implies two consequences. Firstly, a user can define any aspect he or she considers important. Users need to specify parameters before they can be used. However, the effort for this can be reduced by creating sets of commonly used parameters that can be offered, obtained and incorporated by interested users. The second fact implied is that DDM is not restricted to decisions concerning service deployment or content retrieval. It can rather be applied flexibly to any decision making process that can formally be described in terms of service, decision, and adaptable parameters. Nevertheless, this statement is only a theoretical conclusion which we have not proved practically by now.

As result of our research we contributed an algorithm for solving the DDM problem. Considering the parameters and metrics we defined, and the assumptions we stated, this algorithm is a straightforward calculation of an overall usefulness value for each alternative DDM assesses. Nevertheless, according to the principle of Occam’s razor [105], simplicity is not a fault as long as the algorithm fulfils the goal of selecting one out of many alternatives that fits best to user preferences.

The DDM algorithm itself is independent of underlying technologies like web services, WSDL for service description purposes, or dedicated service discovery mechanisms, because we avoided using technology
An algorithmic approach to service and content deployment decision making

specific functionality. However, to apply the algorithm it needs to be implemented. We put into practice a prototype based on the Java programming language and XML. We added example scenarios as well as example parameters. According to the application domain a DDM user may envisage, web services, CASCADAS ACEs, or other technology might be advantageous.

By analysing the algorithmic complexity of the DDM algorithm, we deduced the factors that influence the effort needed to calculate deployment decisions. We showed that it has polynomial worst case runtime if interpreters of adaptable parameters are implemented efficiently. Afterwards, we explained that applying the principle of maximum attainable usefulness (MAU) can lead to considerable savings regarding the average case runtime without a loss in quality of the calculated solution. Further research has to clarify to which extent this principle can be generalised to other algorithmic problems.

Throughout our work, we examined several issues concerning DDM. Any aspect we addressed was evaluated thoroughly either by analytical or by experimental investigation. In terms of theoretical analyses, we built upon commonly accepted fundamentals (like complexity theory for determining the algorithmic runtime of DDM). We clarified the validity of our evaluations by making assumptions and explaining their necessity. In terms of experimental evaluations, we utilised simulations which we designed artificially according to the aspect under investigation. Moreover, we set up experiments which involved existing content providers. We queried these providers using publicly available interfaces and utilised offered content metadata to derive service parameters for describing offered alternatives. The entirety of our experiments that involved real world providers and data is of limited scope. It is not sufficiently extensive to conclude general properties and conditions of the applicability of DDM. Nevertheless, we gained a first insight into potential and limitation of using the presented deployment decision making approach in existing real world domains. It turned out that, for example, video content is available and accessible. Moreover this content is enhanced with multiple metadata that can be used to describe content retrieval alternatives in terms of service parameters. That means that DDM can be applied to fulfil an according retrieval decision task.

Altogether, we investigated a dedicated problem and developed a solution with regard to demands we imposed as well as assumptions made. We examined various aspects of the proposed solution analytically, based
on simulations, and based on real world experiments. The results of our research and the findings we achieved were contributed to the scientific discourse by publishing articles and presenting at scientific conferences and workshops. We invite others to apply the work portrayed in this thesis, to adapt it to further requirements, or to gain ideas for future activities.

6.3 Further work

Along the way of developing a solution for the DDM problem, we determined several further issues. We mentioned them in the relevant sections of this thesis to point out the need for additional investigation. These open issues, summarised in the following paragraphs, may inspire readers to continue researching autonomic, service oriented systems and deployment decision making.

In section 4.3.1 we described how parameter selection affects DDM. It was shown that involving subsets of all available parameters into a deployment decision reduces the computational effort needed at the cost of quality of the calculated solution. A facility was presented which enables the user to select decision and adaptable parameters. Service parameters are denoted by service and content providers. For the time being, no facility exists to delimit their amount. Future work may lead to developing a mechanism that is capable of selecting service parameters for each assessed alternative, in a meaningful and efficient manner.

When analysing the algorithmic complexity of DDM, it turned out that the process of matching service and decision parameters influences the computational effort required. For the time being, we perform this matching in a straightforward manner by comparing each decision parameter to each service parameter for each alternative until a match is found (see section 4.4.1). Developing a more advanced strategy might lead to runtime savings and increase the efficiency of DDM.

In section 4.4.1 we introduced the principle of maximum attainable usefulness (MAU) which helps saving runtime for average case input data. To investigate the behaviour of MAU, we carried out a simulation in which we initialised service parameters in a way that leads to a uniform probability distribution of interpreted decision parameter values. Future work may clarify if different probability distributions are more representative in further scenarios, and in which way the distribution applied influences the average case runtime gain.
The results of our investigation into how MAU improves the runtime performance of DDM showed clear savings within the scope of the scenario we considered. To be able to state more general assertions about the potential of MAU, future work might determine analytically a relation between input data and runtime savings achieved when MAU is applied. The mathematical proof of such a relation would enable potential savings for similar algorithmic problems to be deduced.

Section 5.2 describes self-adaptability capabilities of DDM. We explained and evaluated how, for example, provider reputation and success rate values can be learned and used to address the problem that the information DDM processes might be unreliable. Developing more advanced reputation mechanisms can be a step towards better handling of unreliable information. Such mechanisms may exchange experiences within a community of distributed DDM systems, or they may incorporate data provided by a trustworthy entity. Additional adaptable parameters that, for example, observe trends of measured service parameter values are imaginable as well. Any way of improving the reliability of information can lead to better acceptance of DDM and to more useful deployment decisions.
Bibliography


An algorithmic approach to service and content deployment decision making


An algorithmic approach to service and content deployment decision making


An algorithmic approach to service and content deployment decision making


An algorithmic approach to service and content deployment decision making


Index

-A-
adaptable parameter .......................................................... 57, 63, 120
ADDO project ........................................................................ 48
ADDOaction project .............................................................. 48
advertisement ................................................................. 23
algorithm ............................................................................ 81
algorithmic complexity .................................................... 28, 81
alternative ................................................................. 27, 57
Android ................................................................................. 48
application domain ........................................................... 27
artificial intelligence .......................................................... 35
autonomic ................................................................. 28
autonomic communication element ............................... 47
autonomic computing ....................................................... 36
autonomic systems ........................................................... 30
average case runtime ..................................................... 82

-B-
broker ................................................................................. 24, 46, 49

-C-
CASCADAS project .......................................................... 47
cloud computing .............................................................. 53
An algorithmic approach to service and content deployment decision making

common parameter ..................................................................................... 64
communication technology ........................................................................ 31
complexity theory ....................................................................................... 82
computational resources ........................................................................... 81
consumer ............................................................................................... 27, 56
content .................................................................................................. 27, 55
content retrieval .................................................................................... 30, 45
context ........................................................................................................ 24

-D-
DARPA agent markup language for services ............................................. 45
decision parameter ................................................................................ 57, 62
deployment decision making ................................................................ 27, 55
device.......................................................................................................... 57

-E-
ecosystems .................................................................................................. 23
entity ........................................................................................................... 56
experience ........................................................................................... 59, 120
extensible markup language........................................................................ 43

-F-
fundamentals ............................................................................................... 35

-H-
hypertext transfer protocol.......................................................................... 44

-I-
IEEE service cup ........................................................................................ 52
IEEE web service challenge ................................................................. 51
internet ........................................................................................................ 26
interpreter ............................................................................................... 115

-J-
Java ............................................................................................................. 48
Java Micro Edition ...................................................................................... 48

-L-
Landau-notation .......................................................................................... 82

-M-
maximum attainable usefulness .................................................................. 99
methodology ............................................................................................... 29
mobile virtual centre of excellence ............................................................. 45
multimedia content ..................................................................................... 25
music video ................................................................................................. 25

-N-
etwork ................................................................................................. 27, 55

-O-
ontology ...................................................................................................... 44

-P-
parameter selection ................................................................................... 114
parameter selection method ........................................................................ 88
parameter weight ...................................................................................... 118
personal assistant agent ............................................................................... 46
personal content manager ........................................................................... 46
service level agreement ........................................................................................................... 41
service oriented architecture ................................................................................................. 39
service parameter .................................................................................................................. 57, 62
software deployment .............................................................................................................. 45
source device .......................................................................................................................... 57
state of the art .......................................................................................................................... 32
success rate ............................................................................................................................ 126
supportive system ................................................................................................................... 36

target device ........................................................................................................................... 57
Times Square ............................................................................................................................ 23

ubiquitous computing ............................................................................................................ 30, 36
uniform cost model .................................................................................................................. 84
uniform probability distribution ............................................................................................. 75
uniform resource locator ......................................................................................................... 44
universal description, discovery, and integration protocol .................................................. 42
unreliable information ............................................................................................................ 28, 120
upper bound ............................................................................................................................ 82
usefulness ................................................................................................................................ 57
user-adaptability ...................................................................................................................... 113

web ontology language for services ...................................................................................... 45
web service .............................................................................................................................. 40
web service description language ......................................................................................... 43
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>web service ontology language</td>
<td>44</td>
</tr>
<tr>
<td>weight</td>
<td>63</td>
</tr>
<tr>
<td>world wide web consortium</td>
<td>40</td>
</tr>
<tr>
<td>worst case runtime</td>
<td>82</td>
</tr>
</tbody>
</table>
Today’s computer networks, like the internet or intranets of companies, interconnect a large number of devices as well as persons. Within such networks, content can be exchanged and services can be utilised. The more complex such networks become, the more opportunities might exist to choose a desired service or content. If multiple of those choices need to be made periodically or in short time, manual selection may not be a suitable procedure. From this fact the question arises, how the service or content selection process can be managed efficiently.

In this book, the author Rico Kusber investigates scientifically the process of selecting services or content for deployment on networked computing systems. He presents an algorithmic approach how services and content can be selected automatically in accordance with user preferences and needs. The book addresses scientists and developers in the fields of Autonomic Computing, Ubiquitous Computing, as well as Service and Content Management.

Rico Kusber, Dipl.-Inf.: Researcher at the University of Kassel, Chair for Communication Technology (ComTec); PhD in Computer Science at the University of Kassel in 2010; Degree in Computer Science at Chemnitz University of Technology in 2006, specialisation in Artificial Intelligence