TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Umweltwissenschaften, Fachrichtung Hydrowissenschaften

Institut für Siedlungs- und Industriewasserwirtschaft, Professur Siedlungswasserwirtschaft

Ontology based model framework for conceptual design of treatment flow sheets

THILO KOEGST

Herausgeber:

Prof. Dr. Peter Krebs

Institut für Siedlungs- und Industriewasserwirtschaft

Technische Universität Dresden

Dresden, Dezember 2013

Dissertation zur Erlangung des akademischen Grades Doktoringenieur
(Dr. Ing.) an der Fakultät Umweltwissenschaften
der Technischen Universität Dresden

vorgelegt von Dipl.-Ing. Thilo Koegst

Gutachter:
Prof. Dr. Peter Krebs
Technische Universität Dresden, Germany
Institut für Siedlungs- und Industriewasserwirtschaft

Prof. Dr.-Ing. habil. Jens Tränckner
Universität Rostock, Germany
Stiftungsprofessur für Wasserwirtschaft

Prof. Dr. ir. Ingmar Nopens
Universteit Gent, Belgium
Department of Mathematical Modelling, Statistics and Bioinformatics

1. Auflage 2013

Institut für Siedlungs- und Industriewasserwirtschaft
Technische Universität Dresden
D-01062 Dresden
ISSN 1615-083X

Diese Publikation (Innenteil) wurde auf Recyclingpapier gedruckt.

Druck: Druckerei und Verlag Hille
Redaktion: Thilo Koegst

Preface

Wastewater is purified in treatment plants with a combination of processes such that effluent concentrations are below threshold values. For communal wastewaters standard process-chains are established, which may vary depending on boundary and operation conditions. For the treatment of industrial wastewaters the variety of process-chains is more diverse, since the composition of the wastewater and with that the succession and the coupling of unit processes are differing substantially depending on the branch of industry. The succession of process-chains is thus often based on experience and conventional assembly of unit processes and respective technologies, without systematic examination of all sorts of combination options.

This unsatisfactory situation is the starting point of the work of Thilo Koegst. He is aiming to develop diverse options of process-combinations and systematically examine and assess them by means of ontologies. Ontology tools are not designed for these kinds of applications. With an ontology, complex systems can be represented without quantification of their intra-relations. Therefore, Thilo Koegst had to achieve major developments in order to make internal quantification possible with an ontology and to be able to deliver a model-based description of biochemical processes.

With his approaches and developments, Thilo Koegst has succeeded in developing a PhD thesis away from the mainstream. Even though the product is not yet ready for application in practice, he could still demonstrate that his approach basically works and, moreover, he was able to program a nucleus of such a tool which he tested for some exemplary cases.

With this first step, the work exhibits the potential of options, which has never been thought of so far. On a long-term perspective, I assume that the employment of such a tool in planning and lay-out becomes possible and might result in a diversification of the spectrum of process-chains. Only after some more studies and development work has been performed, we will be able to realistically judge how these methodological findings can be evaluated and whether we can expect an added value in terms of process engineering.

The work was carried out in the frame of several funded research projects. We greatly acknowledge the funding of German Federal Ministry of Education and Research (BMBF) and of EU DG Environment.

Peter Krebs

Vorwort

Abwasser wird in Reinigungsanlagen mit einer Kombination von Verfahren soweit gereinigt, dass die Ablaufkonzentrationen Grenzwerte unterschreiten. Für kommunale Abwässer haben sich dazu Standard-Verfahrensketten etabliert, die je nach Rand- und Betriebsbedingungen variieren können. Für die Behandlung von Industrieabwasser ist die Vielfalt der Verfahrensketten größer, weil sich die Abwasserzusammensetzungen und folglich die Abfolgen und Verschaltungen von Prozessmodulen je nach Industriezweig deutlich voneinander unterscheiden. Die Zusammenstellung von Verfahrensketten basiert deswegen häufig auf Erfahrungen bzw. auf üblichen Technologieabfolgen, ohne dass eine systematische Prüfung verschiedenster Kombinationsmöglichkeiten erfolgt. Thilo Koegst nahm diese unbefriedigende Situation als Ausgangspunkt für seine Arbeit. Er setzte sich zum Ziel, mittels einer Ontologie verschiedenste mögliche Verfahrenskombinationen systematisch zu prüfen und zu bewerten. Ontologien sind nicht primär für derartige Anwendungen vorgesehen. Eine Ontologie erlaubt die formalisierte Abbildung komplexer Systeme, ohne Quantifizierung der Zusammenhänge. Herr Koegst musste deswegen wesentliche Entwicklungsarbeit leisten, um der Ontologie "die Quantifizierung beizubringen" und biochemische Prozesse und deren Leistung modellbasiert zu beschreiben.

Es ist ihm mit seinen Entwicklungen und Herangehensweisen gelungen, eine Arbeit weit abseits des Mainstreams zu liefern. Auch wenn noch kein praxisbereites Werkzeug zur Verfügung steht, so konnte Herr Koegst doch nachweisen, dass seine Herangehensweise grundsätzlich möglich ist, und zudem konnte er bereits einen Kern eines derartigen Werkzeugs programmieren und an Hand einiger Fallbeispiele testen.

Mit diesem ersten Schritt konnte ein Potenzial an Möglichkeiten aufgezeigt werden, die bisher noch nirgends angedacht wurden. Mittel- und langfristig gehe ich davon aus, dass die Verwendung eines ähnlichen Werkzeugs in der Planung und Auslegung sehr wohl möglich ist und zu einer Erweiterung des Spektrums an Verfahrensketten führen kann. Erst nach weiteren Arbeiten in diese Richtung werden wir aber verlässlich einordnen können, wie der Wert dieser methodischen Erkenntnisse zu bewerten und ob verfahrenstechnisch ein Mehrwert zu erwarten ist.

Die Arbeit ist im Rahmen mehrerer geförderter Forschungsprojekte entstanden. Wir danken dem BMBF und der EU Forschungsförderung dafür herzlich.

Peter Krebs

Danksagung

Rückblickend lässt sich feststellen, dass es zwei Dinge gibt, die man zu Beginn einer Promotion wissen, beziehungsweise nicht wissen sollte.
Erstens, ist es eine große Erleichterung nicht von Anbeginn alle Probleme zu kennen und zu durchschauen, die man im Laufe der Promotion zu bewältigen hat, andernfalls wäre man möglicher Weise geneigt, gänzlich von diesem Weg abzusehen. Dies gilt sicher für jede größere Unternehmung.
Zweitens, ist es gut zu wissen von einer Vielzahl von Unterstützern umgeben zu sein, die einem mit Rat und Tat zur Seite stehen und begleiten. Zu diesen gehören vor allem meine Betreuer. Bedanken möchte ich mich bei Peter Krebs, der es mir ermöglicht hat meine Promotion überhaupt zu schreiben, für seine Geduld auf meinem ausgedehnten, lehrreichen Weg von der Idee zur Umsetzung meines Promotionsthemas. Mein Dank geht auch an Jens Tränckner, der mich mit seiner offenen und pragmatischen Art immer wieder neu motiviert und bis zum Abschluss meiner Arbeit begleitet hat. Vielen Dank auch an Ingmar Nopens, der mit kritischen Fragen und Anmerkungen, sowie mit detaillierten Korrekturen meine Arbeit erfolgreich gefördert hat.
Bedanken möchte ich mich auch bei meinen Kollegen am Institut für Siedlungswasserwirtschaft. Zahlreiche Vorträge über den Sinn und Unsinn von Ontologien mussten sie über sich ergehen lassen, und haben mich dennoch mit hilfreichen Hinweisen unterstützt. Ganz besonders danken möchte ich Frank Blumensaat, meinem langjährigen Mitbewohner im Büro, Danke für die vielen anregenden Diskussionen, und überhaupt die gemeinsame Zeit. Ebenso wichtig war mir die Unterstützung von Michael Brodien, Nora Schindler, Sebastian Kempke, Christian Karpf, Björn Helm und Jörg Seegert als langjährige Weggefährten.
An dieser Stelle möchte ich mich auch bei den ehemaligen Informatikstudenten Wolfgang Bücke und Jörg Eichhorn bedanken. Erst sie haben mir den Zugang zur Programmierung und den Umgang mit Ontologien eröffnet.
Nicht zuletzt möchte ich meinen Eltern danken für Ihre Geduld und unschätzbare Unterstützung und Ermutigung. Zweifellos der größte Dank gebührt meiner lieben Frau Tanya.

Thilo Koegst

Abstract

The primary objective of wastewater treatment is the removal of pollutants to meet given legal effluent standards. To further reduce operators costs additional recovery of resources and energy is desired by industrial and municipal wastewater treatment. Hence the objective in early stage of planning of treatment facilities lies in the identification and evaluation of promising configurations of treatment units. Obviously this early stage of planning may best be supported by software tools to be able to deal with a variety of different treatment configurations.

In chemical process engineering various design tools are available that automatically identify feasible process configurations for the purpose to obtain desired products from given educts. In contrast, the adaptation of these design tools for the automatic generation of treatment unit configurations (process chains) to achieve preset effluent standards is hampered by the following three reasons.

First, pollutants in wastewater are usually not defined as chemical substances but by compound parameters according to equal properties (e.g. all particulate matter). Consequently the variation of a single compound parameter leads to a change of related parameters (e.g. relation between Chemical Oxygen Demand and Total Suspended Solids). Furthermore, mathematical process models of treatment processes are tailored towards fractions of compound parameters. This hampers the generic representation of these process models which in turn is essential for automatic identification of treatment configurations.

Second, treatment technologies for wastewater treatment rely on a variety of chemical, biological, and physical phenomena. Approaches to mathematically describe these phenomena cover a wide range of modeling techniques including stochastic, conceptual or deterministic approaches. Even more the consideration of temporal and spatial resolutions differ. This again hampers a generic representation of process models.

Third, the automatic identification of treatment configurations may either be achieved by the use of design rules or by permutation of all possible combinations of units stored within a database of treatment units. The first approach depends on past experience translated into design rules. Hence, no innovative new treatment configurations can be identified. The second approach to identify all possible configurations collapses by extremely high numbers of treatment configurations that cannot be mastered. This is due to the phenomena of combinatorial explosion. It follows therefrom that an appropriate plan-

ning algorithm should function without the need of additional design rules and should be able to identify directly feasible configurations while discarding those impractical.

This work presents a planning tool for the identification and evaluation of treatment configurations that tackles the before addressed problems. The planning tool comprises two major parts. An external declarative knowledge base and the actual planning tool that includes a goal oriented planning algorithm. The knowledge base describes parameters for wastewater characterization (i.e. material model) and a set of treatment units represented by process models (i.e. process model). The formalization of the knowledge base is achieved by the Web Ontology Language (OWL).

The developed data model being the organization structure of the knowledge base describes relations between wastewater parameters and process models to enable for generic representation of process models. Through these parameters for wastewater characterization as well as treatment units can be altered or added to the knowledge base without the requirement to synchronize already included parameter representations or process models. Furthermore the knowledge base describes relations between parameters and properties of water constituents. This allows to track changes of all wastewater parameters which result from modeling of removal efficiency of applied treatment units.

So far two generic treatment units have been represented within the knowledge base. These are separation and conversion units. These two raw types have been applied to represent different types of clarifiers and biological treatment units.

The developed planning algorithm is based on a Means-Ends Analysis (MEA). This is a goal oriented search algorithm that posts goals from wastewater state and limit value restrictions to select those treatment units only that are likely to solve the treatment problem. Regarding this, all treatment units are qualified according to postconditions that describe the effect of each unit. In addition, units are also characterized by preconditions that state the application range of each unit. The developed planning algorithm furthermore allows for the identification of simple cycles to account for moving bed reactor systems (e.g. functional unit of aeration tank and clarifier). The evaluation of identified treatment configurations is achieved by total estimated cost of each configuration.

The planning tool has been tested on five use cases. Some use cases contained multiple sources and sinks. This showed the possibility to identify water reuse capabilities as well as to identify solutions that go beyond end of pipe solutions.

Beyond the originated area of application, the planning tool may be used for advanced interrogations. Thereby the knowledge base and planning algorithm may be further developed to address the objectives to identify configurations for any type of material and energy recovery.

Keywords: Ontology, model representation, industrial wastewater treatment, conceptual design, planning, treatment flowsheets

Kurzfassung

Das primäre Ziel der Abwasserreinigung besteht in der Entfernung von Schmutzstoffen, bis vorgegebene Grenzwert-Konzentrationen zur Abwassereinleitung erreicht sind. Zur Reduktion von Betriebskosten wird darüber hinaus sowohl für industrielle als auch für kommunale Abwasserreinigung eine Stoff- und Energierückgewinnung angestrebt. Die Aufgabe zu Beginn der Planung von Behandlungsanlagen besteht daher in der Identifikation und anschließenden Bewertung von sinnvollen Verfahrenskonfigurationen. Es liegt nahe für die anfängliche Identifikation und Bewertung Software gestützte Werkzeuge zu nutzen, um eine große Anzahl von Verfahrenskonfigurationen berücksichtigen zu können.

In der chemischen Verfahrenstechnik existieren schon länger derartige Werkzeuge zur Planung von Verfahrensketten. Das Ziel besteht dabei in der Identifikation von Prozessketten, um aus gegebenen Edukten gewünschte Produkte zu erzeugen. Eine unmittelbare Anwendung dieser Werkzeuge für die Planung von Verfahrenskonfigurationen zur Abwasserbehandlung ist nicht möglich. Dies ist im Wesentlichen auf drei Gründe zurückzuführen.

Erstens, die Charakterisierung von Abwasserqualität erfolgt nicht ausschließlich über chemische Substanzen, sondern oft über Gruppen- und Summenparameter. Diese überschneiden sich in ihren Aussagen, so dass die Änderung eines Parameters auch Auswirkungen auf weitere Parameter haben kann (z.B. die Beziehung zwischen partikulären Stoffen und Chemischem Sauerstoff-Bedarf). Des Weiteren basiert die mathematische Prozessmodellierung von Behandlungsprozessen oft nicht auf der Beschreibung von Summen- und Gruppenparametern, sondern aus davon abgeleiteten Fraktionen. Dies erschwert eine generische Beschreibung von Prozessmodellen, die aber für eine automatische Identifikation von Verfahrensketten nötig ist.

Zweitens, Technologien zur Abwasserbehandlung basieren auf einer Vielzahl chemischer, biologischer und physikalischer Phänomene. Die modelltechnischen Ansätze zur Beschreibung dieser Phänomene sind sehr heterogen und umfassen konzeptionelle, deterministische bis hin zu stochastischen Ansätzen. Ferner variieren diese Ansätze auch hinsichtlich ihrer räumlichen und zeitlichen Charakterisierung. Diese Besonderheit erschwert die Formulierung einer Modelldatenbank zur Beschreibung von Behandlungsverfahren als Voraussetzung zur automatischen Identifikation und Bewertung von Prozessketten.

Drittens, die automatische Identifikation von sinnvollen Verfahrenskombinationen kann entweder durch die Vorgabe von Entwurfs-Regeln erfolgen oder durch die Erzeugung von Verfahrenskonfigurationen als Ergebnis aller möglichen Kombinationen von vorhandenen Behandlungs-Verfahren einer Modelldatenbank. Der erste Ansatz beruht auf zurückliegenden Erfahrungen und ermöglicht keine souveräne Identifikation von neuen Verfahrensketten. Der zweite Ansatz führt sehr schnell zu einer großen, nicht beherrschbaren Anzahl von Verfahrensketten durch das Phänomen der kombinatorischen Explosion. Ein brauchbarer Planungsalgorithmus sollte daher ohne zusätzliche Entwurfs-Regeln funktionieren und nur solche Verfahrensketten identifizieren die für eine anschließende Bewertung sinnvoll erscheinen.

In dieser Arbeit wird ein Planungswerkzeug zur Identifikation und Bewertung von Verfahrenskombinationen zur Abwasserbehandlung vorgestellt, welches die zuvor adressierten Probleme löst. Das Planungswerkzeug besteht aus zwei wesentlichen Teilen: einer externen deklarativen Wissensdatenbank und dem eigentlichen Software-Werkzeug inklusive eines zielorientierten Planungsalgorithmus. Die Wissensdatenbank beschreibt zum einen Parameter zur Abwassercharakterisierung (i.e. *material model*) und zum anderen Verfahren zur Abwasserbehandlung durch deren Prozessmodelle (i.e. *process model*). Die Formalisierung der Wissensdatenbank basiert auf der Web Ontology Language (OWL).

Das entwickelte Datenmodell der Wissensdatenbank beschreibt die Zusammenhänge zwischen Parametern und Prozessmodellen in einer Art und Weise, dass Prozessmodelle generisch beschrieben werden können. Somit können weitere Abwasserparameter und Verfahren zur Abwasserbehandlung unabhängig voneinander geändert und hinzugefügt werden, ohne bestehende Modelle oder Parameter ändern zu müssen. Die Wissensdatenbank beschreibt weiterhin Zusammenhänge zwischen Abwasserparametern. Nach der Berechnung der Reinigungswirkung von Behandlungsverfahren kann somit auf die Auswirkung aller Abwasserparameter geschlossen werden.

Hinsichtlich der Repräsentation von Prozessmodellen sind bislang zwei generische Verfahrensblöcke im Datenmodell implementiert, diese sind Abtrennprozesse und Umsatzprozesse.

Der implementierte Planungsalgorithmus basiert auf einer erweiterten *Means-ends analysis*. Dies ist ein zielorientierter Suchalgorithmus, welcher für Abwasserzustände und Reinigungsgrenzwerte Ziele definiert, die von Verfahren aus der Wissensdatenbank gelöst werden müssen. Verfahren werden daher so

genannte Post-Zustände (i.e. *postcondition*) zugeordnet, die den Nutzten eines jeden Verfahrens beschreiben. Weiterhin können Verfahren nur dann in Verfahrensketten einbezogen werden, wenn diese Vorbedingungen erfüllen (i.e. *precondition*). Der implementierte Planungsalgorithmus erlaubt auch die Berücksichtigung von einfachen Zyklen, wie sie bei der biologischen Abwasserreinigung im Fließbettverfahren üblich sind (d.h. Funktionseinheit von Belebungsbecken und Nachklärbecken). Die Bewertung identifizierter Verfahrensketten, die die vorgegebenen Grenzwerte erfüllen, erfolgt über die Gesamtkosten jeder Verfahrenskette.
Das entwickelte Planungswerkzeug wurde an fünf Beispielen getestet. Dabei wurden auch initiale Pläne untersucht, die mehr als eine Abwasserquelle sowie mehrere Senken beinhalten. Somit konnte gezeigt werden, dass auch Möglichkeiten zur Wasserrückgewinnung identifiziert werden können.
Über die ursprüngliche Nutzung des Planungswerkzeuges hinaus kann die entwickelte Wissensdatenbank und auch der Planungsalgorithmus genutzt werden, um weiter gehende Fragestellungen zu beantworten. In möglichen Weiterentwicklungen des Planungswerkzeuges sollte es möglich sein, Verfahrenskonfigurationen zu identifizieren, die auch Stoff- und Energierückgewinnungsaspekte beinhalten.

Schlagworte: Ontologie, Modell-Repräsentation, Abwasserbehandlung, Generierung von Prozessketten

Contents

Preface i

Vorwort iii

Danksagung v

Abstract vii

Kurzfassung xi

Contents xiv

List of Figures xix

List of Tables xxiii

List of Listings xxiv

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Objective . . . 4
- 1.3 Problem analysis . . . 6
  - 1.3.1 General . . . 6
  - 1.3.2 Requirements for model development . . . 6
  - 1.3.3 Requirements for model application . . . 8

2 Literature Review 11
- 2.1 Introduction . . . 11
- 2.2 Wastewater characterization . . . 12
  - 2.2.1 Introduction . . . 12
  - 2.2.2 Qualitative Parameters . . . 14
  - 2.2.3 Properties of compounds . . . 22

2.3 Treatment options and modeling approaches 26
2.3.1 Introduction 26
2.3.2 Physical phenomena and application in treatment processes 29
2.3.3 Biological Processes 36
2.4 Knowledge Representation and Reasoning 40
2.4.1 Introduction 40
2.4.2 Representing knowledge in Logic 43
2.4.3 Representing knowledge in Ontologies 49
2.4.4 Possibilities, limitations, and trade-offs 59
2.5 Process Modeling 60
2.5.1 Introduction 60
2.5.2 Definitions & Terminology 61
2.5.3 Application of process models 63
2.5.4 Problems and Requirements 64
2.5.5 Modeling Process 66
2.5.6 Modeling Concepts 69
2.5.7 Modeling Languages 74
2.6 Process Design and Optimization 82
2.6.1 Introduction 82
2.6.2 Planning Theory 84
2.6.3 Conceptual Process Synthesis Methods 87
2.6.4 Applications for design of wastewater treatment chains 90
2.7 Summary 95

3 Material and Methods 97
3.1 Introduction 97
3.2 Used resources 98
3.3 Definition of use cases 101

4 Results 105
4.1 Introduction 105
4.2 Assumptions and boundary conditions 109
4.3 Naming conventions and definitions 109
4.4 Layered model representation and auxiliary concepts 111
4.4.1 Introduction 111
4.4.2 Representing algebraic terms 112
4.4.3 Representing conditional terms 113

4.4.4 Class surrogates 114
4.4.5 Units 116
4.4.6 Representing processes 117
4.4.7 Properties 120
4.5 Material model 120
4.5.1 Introduction 120
4.5.2 Concept definition by class taxonomy 121
4.5.3 Individuals and properties 130
4.5.4 Model interpretation and application 130
4.6 Fractionation and Accumulation of states 134
4.6.1 Introduction 134
4.6.2 Fractionation rules and algorithm 137
4.6.3 Accumulation 139
4.6.4 Exemplary fractionation 139
4.7 Process model 141
4.7.1 Introduction 141
4.7.2 Generic model units 145
4.7.3 Postconditions and Preconditions 146
4.7.4 Recycle flow 147
4.7.5 Biological carbon removal (with MBR) 148
4.7.6 Exemplary model units 149
4.8 Flow sheet generation 154
4.8.1 Introduction 154
4.8.2 Implementing state space search 156
4.8.3 Uninformed search 157
4.8.4 Means-Ends Analysis (MEA) 161
4.8.5 Controlling search 167
4.9 Application 170
4.9.1 Introduction 170
4.9.2 Use case #1 and #2 171
4.9.3 Use case #3 176
4.9.4 Use case #4 176
4.9.5 Use case #5 176
4.10 Discussion 181

5 Conclusion **187**
5.1 Summary and research contribution 187
5.2 Outlook and further research 190

Glossary **193**

Nomenclature **199**

Bibliography **201**

A On organic carbon and nutrient removal **217**
A.1 Dimensioning of aeration tanks according to Henze *et al.* (2008) 217
A.2 Modeling organic carbon and nitrogen removal 220

B Propositional Logic and Predicate Logic **223**
B.1 Propositional Logic 223
B.2 Predicate Logic 225

C GUI flowsheet finder **233**

List of Figures

1.1 Model representation and implementation 3
1.2 Required input specification: wastewater characteristics of sources and effluent standards on sinks 4
1.3 Relation between parameters for wastewater characterization (x) and properties of wastewater constituents (y) as required by modeling treatment efficiency 7

2.1 Systematization of wastewater characteristics though the perspective of parameter, properties, and compounds 14
2.2 Equilibrium $NH_4^+ \rightleftharpoons NH_3 + H^+$ ($pKs = 9.25$) 20
2.3 Solids parameters 21
2.4 Spectrum of contaminants regarding particle size range and measurement techniques 22
2.5 Comparison between Particle Size Distribution of municipal wastwater (•= effluent primary treatment; ○= effluent secondary settler) industrial wastewater (•= buffer tank; △= effluent flotation, ○= effluent secondary settler) and wastewater from a pig farm (•= effluent pigsty; ○= effluent wastewater pond) published in Sophonsiri and Morgenroth (2004) 25
2.6 Causal chain of wastewater characteristics, applied phenomenological treatment processes and implemented technology 27
2.7 Relation between wastewater constituents and treatment processes 28
2.8 Application range centrifuge technology (Stahl, 2004) 33
2.9 Effect of influent suspended solids on removal efficiency according to equation 2.11 34
2.10 Application range of pressurized membrane technology (Rautenbach, 1997) 36
2.11 COD Fractions of ASM1 38
2.12 Influences on OWL (adopted from Patel-Schneider 2004) . . . 59

2.13 Evolution of Process Model Representations (adapted from Bieszczad 2000) 65
2.14 Modelling Process (adapted from Wedel *et al.* 2002, p. 90) . . . 66
2.15 Three dimensions of process modelling (Bogusch and Marquardt, 1997) 69
2.16 Information between devices and connections (Mangold *et al.*, 2002; Marquardt, 1995b) 71
2.17 A taxonomy of process quantities (adapted from Marquardt 1995b) 72
2.18 A taxonomy of model equations (adapted from Marquardt 1995b) 72
2.19 Example Production Tree (Bieszczad, 2000) 78
2.20 Expanded Phenomena-Based Model Production Tree (Bieszczad, 2000) 81
2.21 Major concepts in VEDA (Bogusch *et al.*, 2001) 81
2.22 Superstructure network featuring two oxic/anoxic reaction units and one sedimenation unit (Rigopoulos and Linke, 2002) 91
2.23 Logical layer and execution layer (adapted from Linninger and Chakraborty 1999) 93
2.24 Methodology for waste treatment synthesis (adapted from Linninger and Chakraborty 1999) 94

3.1 Overall applied method 97
3.2 Protégé - Entities panel 99
3.3 Protégé - OWL Viz panel 99
3.4 Wastewater treatment chain of a particular brewery (DWA, 2009, p. 165); hydraulic load between 2000 and 3000 m^3/d . . 102

4.1 Program sequence 106
4.2 Components of a state-transition-network (STN) 106
4.3 Separation of model and model application (separation of knowledge and knowledge processing) 107
4.4 Layered model representation 112
4.5 Binary tree representing equation $z = 2x + y^2$ with $a_1 \in$ `Assign`, $t_1 \in$ `Plus`, $t_2 \in$ `Multiply`, $t_3 \in$ `Power`, $\{x, y, z\} \in$ `Variable`; the symbols r, ls, and rs represent the properties `hasResult`, `hasLeftSide`, and `hasRightSide` 114

4.6 Representing the condition $COD < 150$ mg/l with $c_1 \in$ Condition, $lesserThan \in$ Comparator, $150 \in$ Constant; the symbols c, ls, and rs represent the properties hasComparator, hasLeftSide, and hasRightSide 114
4.7 Exemplary use of ClassSurrogate concept 116
4.8 Graph of OWL representation of heterotrophic growth (see table 4.4), with $a_i \in Assign$ and $i_j \in Term$ using the following property-relations: hSt-hasStoichiometric Term, hasRate-hasRate, hR-hasResult, hSPC-hasSuperClass, hLS-hasLeftSide, hRS-hasRightSide 119
4.9 Three categories for material description 121
4.10 Asserted class taxonomy of concept SubstanceProperty . . . 125
4.11 Quality Parameter - Inferred class taxonomy 129
4.12 Relation between class taxonomy of concepts and instances. Classes are indicated by boxes above horizontal line. Class hierarchy is indicated by dashed lines. Individuals are indicated by diamonds below horizontal line. Membership of individuals to classes is indicted by solid lines from classes to individuals. Lines between individuals indicate property relation hasEquivalent. 131
4.13 Interaction between model and model environment to represent the assignment between parameter and amount (e.g. COD=100 mg/l) 133
4.14 Individuals in initial state according to table 4.7 (only quality parameter indicated) 143
4.15 Individuals in fractionated state according to table 4.7; only quality parameter indicated (for legend see figure 4.14) 144
4.16 Graph representation of the ModelUnit concept; *more than one relation possible 145
4.17 Generic unit blocks with inlet and outlet ports; Outlet ports of separation and sink units vary regarding assigned hydraulic status (see table 4.9) 145
4.18 Defining state space search and resulting search tree in the context of flow sheet generation (see example 3) 156
4.19 Example for the execution of task 1 - define search target (see example 4) 159

4.20 Example on closing cycles - position of merge unit is identified through hydraulic status of states (see example 5) 160
4.21 Steps in performing a means-ends analysis 162
4.22 Overview on the implementation of MEA 166
4.23 Minimal configuration of flowsheets for use case #1 and #2. Values on edges represent the flow rate in m^3/h 171
4.24 Use case #1 - evolution of concentrations on outlet ports . . . 172
4.25 Use case #2 - evolution of concentrations on outlet ports . . . 174
4.26 Three examplary results of use case #3 178
4.27 Three exemplary results of use case #4 179
4.28 Three exemplary results of use case #5 (SP - split unit, ME - merge unit) 180
4.29 Options for evaluating list of flowsheets 184

A.1 Notation of fractions in the MBR cycle according to Henze *et al.* (2008) 218

C.1 GUI in setup mode, showing initial plan 234
C.2 GUI showing search results 234
C.3 Preference panel 235
C.4 Export window for identified flowsheets 235

List of Tables

2.1 Systematization of quality parameters according to Koppe and Stozek (1993) 15
2.2 Characterization of compound parameters in municipal wastewater (Koppe and Stozek, 1993, p.34-50) 16
2.3 Relations between nitrogen compounds 18
2.4 Grain classes (Rickert and Hunter, 1971) 23
2.5 Relation between bio-degradability and dispersity on examples (Koppe and Stozek, 1993, p. 53) 24
2.6 Treatment technologies with respect to particle size (Imhoff and Imhoff, 1993, p. 89) 27
2.7 influencing factors on the effectiveness of flotation (Walter, 2005, p. 19) 36
2.8 Representation of stoichiometric matrix for an extensive reaction system (Gujer, 2008, p. 73) 39
2.9 Kinetics and stoichiometrics for carbon removal (Olsson and Newell, 1999, p. 60) 39
2.10 Formal representation languages (Russel and Norvig, 2003, p. 308) 44
2.11 Valid operators in DLs (Angeli, 2007; Brachman and Levesque, 2004) 46
2.12 Syntax and Semantics (Tessaris, 2001) 47
2.13 Types of DLs (Schmidt-Schauss and Smolka, 1991) 48
2.14 the expressivity is encoded by a letter (Schmidt-Schauss and Smolka, 1991) 49
2.15 Various types of ontologies with different degrees of expressiveness 53
2.16 Overview on philosophical notions 57
2.17 Types of modeling tools 63

3.1 Range of concentrations of brewery wastewater in mg/l . . . 101

3.2 Limit values for direct discharge in mg/l (AbwV, 2004) 101
3.3 Definition of sources and sinks, concentrations in g/m^3 and flow in m^3/h 103
3.4 Definition of use cases 103

4.1 The Manchester OWL Syntax, restrictions and boolean class constructs (Horridge *et al.*, 2006) 111
4.2 Terminology for OWL development and OWL design pattern (Horridge *et al.*, 2006, 2009) 111
4.3 Units accounted in the declarative model 116
4.4 Heterotrophic Growth according to Olsson and Newell (1999, p. 58) 118
4.5 properties defined within auxiliary concepts 120
4.6 Sub-concept under the concept `ParameterType` 123
4.7 Exemplary state before and after fractionation, based on fractionation rules as defined in table 4.8 136
4.8 Exemplary rule set involved in fractionating parameter list according to table 4.7 142
4.9 Definition of hydraulic status (hs) 148
4.10 Secondary preconditions of unit types 148
4.11 Overview on exemplary model units - Separation units 151
4.12 Overview on exemplary model units - Conversion units . . . 153
4.13 Term definition regarding flow sheet generation 155
4.14 Term definition regarding MEA 163
4.15 Exemplary generation of a goal and postconditions from a state (see example 6) 164
4.16 Fractionated and accumulated states of sources from table 3.3, only concentrations (g/m^3) and ratios 170
4.17 Use case #1 - outflow concentrations in g/m^3 and flow in m^3/h 172
4.18 Outflow concentrations in g/m^3 and m^3/h 173
4.19 Overview on search results 177

A.1 Modeling parameters Henze *et al.* (2008, p. 35) 218
A.2 Exemplary wastewater concentrations (in mg/l), Henze *et al.* (2008, p. 35) 220
A.3 Kinetics for carbon and nitrogen removal (Olsson and Newell, 1999, p. 67,71) 221

A.4 Kinetics for simple fermentation model (Olsson and Newell, 1999, p. 76) 221
A.5 Used coefficients according to example 3.8 and 4.1 in Olsson and Newell (1999) 222

B.1 Logical connections (i.e. logical junctions) 224
B.2 Truth-table 224
B.3 Logical equivalence 225
B.4 Rules of inference 226
B.5 Logical equivalence 228

Listings

4.1 Result of the recursive parsing of process growth of heterotrophic biomass 118
4.2 RDF/XML rendering of concept COD in the OWL file (based on proposition 4.13) 128
4.3 RDF/XML rendering of concept TKN in the OWL file (based on proposition 4.15) 128
4.4 Method GETAMOUNT of class QUANTITYSET 134
4.5 Method SETAMOUNT of class QUANTITYSET 135
4.6 Fractionation algorithm (method FRACTIONATION of class FRACTIONATION) 140
4.7 Encoding of nitrification unit in OWL(DL) 152

1
Introduction

1.1 Motivation

Conceptual process design[1] in the field of chemical engineering is about finding process chains for transforming matter from inexpensive raw materials to highly desired products while minimizing cost and consumed resources (Harmsen, 2004). According to Douglas (1988, p. xv) the goal of a conceptual process design is to find the best process flow sheet (i.e. to select process units and the interconnections among these units) and estimate the optimum design conditions.

The idea of conceptual process design may have already been revealed as Arthur D. Little first stated the term unit operations in 1915. He noted that[2]: "Any chemical process, on whatever scale conducted, may be resolved into a co-coordinate series of what may be termed Unit Operations, as pulverizing, dyeing, roasting, crystallizing, filtering, evaporation, electrolyzing, and so on. The number of these basic operations is not large and relatively few of them are involved in any particular process. The complexity of chemical engineering results from the variety of conditions as to temperature, pressure, etc., under which the unit operations must be carried out in different processes and from the limitations as to materials of construction and design of apparatus imposed by the physical and chemical character of the reacting substances."

The concept of unit operations can be understood as a convenient manner of organizing chemical engineering knowledge.

Regarding conceptual process design, unit operations have been in focus over a long time. Only later in the 1990s the scale has been expanded to nano and micro scale on the one hand and to macro and mega scale on the other hand (Li and Kraslawski, 2004).

Furthermore, the objective of conceptual process design has been expanded to account for recovery of wasted resources and treatment, and recycle of waste

[1] This terms is sometimes referred to as conceptual process synthesis or simply conceptual design. In this work no distinction will be assumed between these terms.

[2] found in Darton *et al.* (2003, p. 21-22)

streams as it is the focus of this work. Some contributions on conceptual design of treatment flow sheets are presented by Flores Alsina (2008); Freitas *et al.* (2000); Linninger and Chakraborty (1999); Linninger *et al.* (2000a); Roda *et al.* (2000a).

A fundamental difference between process design in its primary idea and conceptual process design of flow sheets for treatment of waste streams lies in the knowledge on constituent materials. In chemical process design raw, intermediate and end products are in general well understood and can be described according to their substance properties, contextual behavior and reaction kinetics. In contrast, in treatment of waste streams pollutants may be recognized by quality parameters, nevertheless a comprehensive substance-specific characterization is commonly not possible. This shortcoming is circumvented by characterizing properties of waste streams (e.g. fraction of particulate settleable matter or fraction of biodegradable matter). Furthermore, properties of waste streams are related to respective treatment methods.

Whatever the objective, todays conceptual design approaches are embedded in the field of Computer-Aided Process Engineering (CAPE). Thereby, conventional chemical process engineering activities dealing with design, construction, and operation of chemical plants are supported by computer based tools (Morbach *et al.*, 2007b). In this perspective, conceptual process design becomes a type of model application. Consequently, knowledge on chemical processes must be formally represented (i.e. the process model) in order to be implemented into a software tool.

The meaning of the term model is thereby ambiguous. In this work, a process model is defined as an abstracted real world phenomenon for a distinct objective. Whenever the objective is to apply a model within a model environment (syn.: software application) the model must be implemented into the model environment. For the comprehension of this work, it is important to introduce the additional term model framework. Thereby a model framework provides the means for model representation. The interdependence of the three concepts of model, model framework, and model environment can be depicted by three interleaved circles (Figure 1.1). The inner most circle represents the model itself being the abstraction of part of the world. The second inner circle represents the model framework which offers a formal language and concepts to represent the model and thereby the model functionality. The outer most circle depicts the model environment wherein the model is actually applied. When talking about a model one may refer to the sole mind model or an al-

ready formalized or implemented one, e.g. a mathematical equation. In most cases, the type of language for model representation is dictated by the model environment. It follows therefrom that some objectives of the actual model may get lost in the process of model representation and final implementation by the model framework.

A stringing necessity of most process models is the consideration of conservation of mass and energy. Such a model functionality (syn.: model behavior) can be accounted for by simple or complex types of mathematical equations. In contrast, it has been argued by various authors (Bieszczad 2000; Linninger *et al.* 2000b; Marquardt 1995b to name only a few), that a sole mathematical representation of process models is not sufficient to deal with real world phenomena. Whereas most conventional approaches focus on behavioral model description only, advanced approaches include additional representation to account for structural and material aspects. In chemical engineering scientific approaches of this kind are referred to as phenomenological-based modeling approaches (Arizmendi-Sanchez and Sharratt, 2008; Bieszczad, 2000; Marquardt, 1995b). Instead of a sole mathematical model formulation phenomenological-based modeling approaches strive for knowledge-based model representations. Especially the incorporation of a profound material view is important for process modeling. Here, materials (e.g. substances, molecular structures) are related to the phase system (contextual behavior), and reaction types (Linninger and Stephanopoulos, 1998; Yang *et al.*, 2003). Through the material view, assertions on properties of substance can be made according to the present boundary conditions (e.g. temperature, pressure, etc.).

For the domain of chemical process engineering Marquardt *et al.* (2010) and Morbach (2009) have presented a reusable Ontology for Computer-Aided Process Engineering (OntoCAPE). Within OntoCAPE necessary concepts are organized in different layers describing interactions between concepts of materials, mathematical models and process units to name only a few. The applied model framework of OntoCAPE is the Web Ontology Language (OWL).

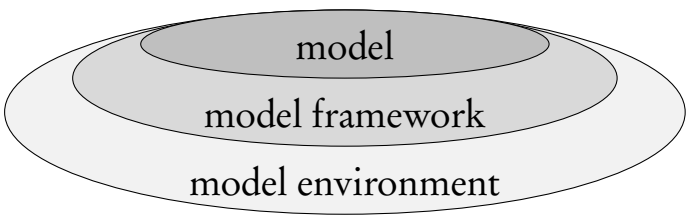


Figure 1.1: Model representation and implementation

In the field of modeling of wastewater systems, in particular the modeling of the activated sludge process, a type of model framework has been already presented by Henze *et al.* (1987) and is referred to as ASM (Activated Sludge Model) approach. Thereby processes are represented by reaction kinetics and stoichiometry and are organized within a equation-matrix. Reactants are derived from fractions of biodegradable matter, active organisms and other constituents. Since then the matrix notation has been successfully used to further introduce new processes to model different unit operations of biological wastewater treatment. Over the past decades the ASM approach has proven to be a sophisticated framework for modeling activated sludge processes. Nevertheless a transfer for the incorporation of other processes beyond biological processes is not easily achieved, especially if a formulation beyond mathematical notation is desired.

Consequently, there are no standardized methods for a generic representation of treatment units aiming at a phenomenological based process description. The scope of this work is therefore the introduction of a model framework to represent a holistic model of treatment units and involved materials.

1.2 Objective

The overall objective of this work is the development of a model framework and environment for conceptual design of treatment configurations for industrial wastewater treatment. The package of model framework and environment is from now on referred to as planning tool. The case of application of the planning tool is the primary development phase of a wastewater treatment facility, that is to assess possible treatment alternatives. These identified alternatives are the basis to finally identify an optimal treatment concept through expert knowledge which in turn will be further developed and realized. The expected advantage of such a preliminary identification of alternatives lies in a fast evaluation of many possible options and possibly reveal alternatives which are beyond standard solutions. Thereby it is regardless if the objective is the development of a fully new treatment facility or the retrofit of an existing one perhaps to implement water reuse or to adapt an existing facility to increased effluent standards.

The required input specifications are a list of wastewater characteristics of one or more wastewater streams as well as a list of one or more possible discharge points (e.g. sewer discharge, sludge disposal, water reuse) with associated effluent standards (Figure 1.2). Given these boundary conditions the planning

tool is expected to identify feasible treatment configurations ranked by a rough cost estimation of each treatment scheme.

Possible treatment units to be considered in the process of flow sheet generation are expected to be queried from a knowledge base. The knowledge base must allow for appending of additional treatment units. The advantage of this feature is the ability to add new treatment concepts and thereby to improve the overall applicability. Furthermore, the appending of already realized reactor configurations offers the evaluation of reuse of these units to new treatment sites. Thereby tedious and expensive redevelopment may be prevented.

The following two important simplifications are admitted to the requirements of the planning tool. The provided wastewater characteristics may only account for steady state flow. More detailed considerations such as dynamic simulation may only be necessary for detailed investigations and hence may be neglected in early stages of planning. Second, the considered degree of detail is defined by those of unit operations. A unit operation may be envisioned as a reactor wherein processes provoke changes to the wastewater state of the inflowing waste stream. In turn, the processed waste stream may be fed to a subsequent unit operation.

The only required user interaction lies in the specification of one or more wastewater sources and one or more sink units. It is at hand that a very detailed specification of wastewater characteristics will yield more relevant treatment trains rejecting those irrelevant. Oppositely, from a sparse input specification one can only expect to yield a broad overview on possible treatment options.

The above formulated objectives are embedded into a profound scientific objective namely to develop a model framework that is capable not only to represent model functionality but also to add contextual information to process models such as boundary conditions and associated material concepts. The objective is furthermore to utilize the concept of ontology for knowledge representation. The resulting knowledge base (i.e. ontology) may be application

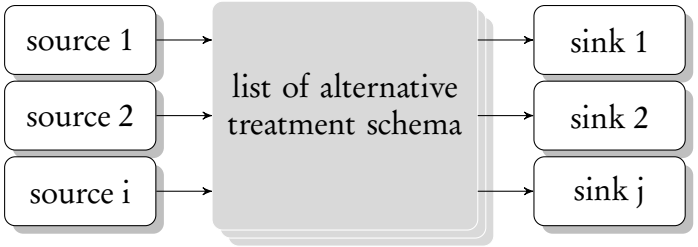


Figure 1.2: Required input specification: wastewater characteristics of sources and effluent standards on sinks

oriented for the particular use to the above introduced planning tool. As a scientific contribution the development and usage of the developed ontology may serve as feasibility study and example for possible other subsequent applications.

1.3 Problem analysis

1.3.1 General

The realization of the contrived objectives is confronted by issues in the areas of model development and model application. Thereby the term model first refers to formalized assertions on the state of a wastewater stream and second to formalized assertions on a treatment unit regarding the ability to induce changes of state, associated boundary conditions and cost. Therefore, model development deals with what is represented and how it is represented. The question of how the model must be represented is associated with the development of a sophisticated model framework to bridge the gap between model objective and its actual application. The area of model application is engaged with the formulation of a planning problem to finally derive feasible treatment trains, from now on complete plans. In this context a complete plan is a set of states, and state transition operations, and its status is complete when the overall plan does not violate preset conditions.
As model development is concerned with the set up and exertion of a model framework, model application is restricted to the developed model environment.
As consequence of the objective to achieve a generic model description that allows at the same time detailed case-specific conclusions, it is important to conserve a strict separation between knowledge on the domain of concern (i.e. the knowledge base) and the functionality required for knowledge processing within the model environment. For the technical realization it follows therefrom that at best no domain knowledge is implemented directly into the model environment.

1.3.2 Requirements for model development

As stated above the domain of concern covers treatment units for wastewater purification and involved wastewater constituents. The model objective is to describe treatment units according to their efficiency to provoke state changes of inflowing waste streams, limitations of use and cost functions. To achieve a most generic unit description, a phenomena-based process specification is necessary.

As noted in section 1.1 parameters for wastewater characterization are not necessarily substance specific. In fact most important parameters are typed "composite parameter" since constituents are combined according to a single particular property. For example, a measured concentration of Chemical Oxygen Demand (COD) summarizes the oxygen equivalent of organic water constituents or a fraction of according to a defined chemical analysis. In contrast, the parameter Volatile Suspended Solids (VSS) represents the matter equivalent of organic undissolved substances. The assertions of both parameter overlap and complement each other since in both cases organic matter is referenced. Given an additional concentration of the substance ethanol it can be assumed that the amount of this substance contributes to COD but not to VSS as ethanol occurs dissolved in liquid phase at a common temperature range. The example above describes dependencies between a quality parameter and properties of a wastewater state as depicted in figure 1.3. In figure 1.3 quality parameters (e.g. COD, VSS, etc.) are represented by $x_{1,2,...,i}$ and properties (e.g. bio-degradable, particulate, etc.) are represented by $y_{1,2,...,j}$. A wastewater state is then a set of individuals. Each individual can be assigned one or more parameters and respective properties.
From the above, it follows therefrom that a given wastewater characterization must be translated to the underlying property distribution of a waste stream. This translation process will be referred to as fractionation. In contrast to a fractionated waste stream, assertions on the actual values of intrinsic quality parameters must be assured (i.e. accumulated waste stream). This is because limit values based on legal regulations are based on standard quality parameters. For municipal wastewater, standardized rules for the fractionation of wastewater parameters have emerged from long term experience. This holds in particular for the fractionation of COD for the modeling of the activated sludge process based on ASM approaches. For industrial wastewater from dif-

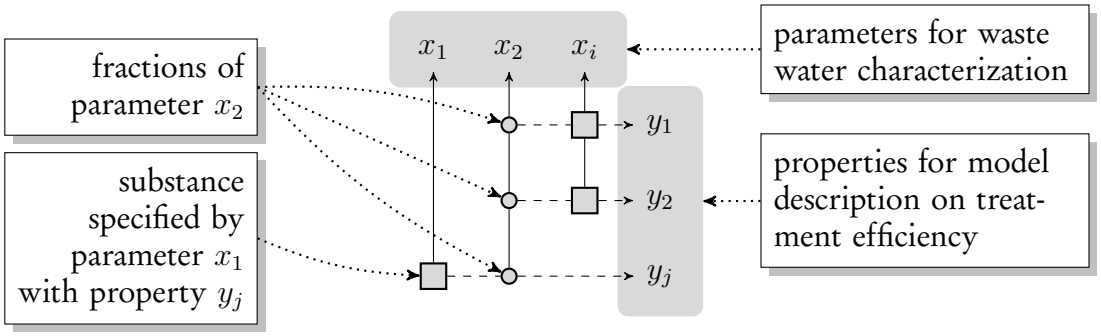


Figure 1.3: Relation between parameters for wastewater characterization (x) and properties of wastewater constituents (y) as required by modeling treatment efficiency

ferent production sites or industrial branches profound insight into wastewater properties can only be achieved by laborious measurement campaigns or expert knowledge. For the scope of this work knowledge on fractionation will be derived from: (i) sophisticated representation of interactions between quality parameters; (ii) contextual description of substances; and (iii) provision of industry specific standard values. The actual fractionation process must be realized as a rule based procedure which allows for the incorporation of the above listed sources of information.
Besides the representation of waste streams as properties of wastewater constitutes it is advisable to organize these properties within a hierarchical system of sub and super type relations. An exemplary treatment unit which affects particulate matter may also consider objects of type suspended or settleable matter. Such an extended property representation contributes to a more generic treatment unit description.

1.3.3 Requirements for model application

The objective regarding model application is to use the above specific model (i.e. ontology) in combination with user specified wastewater characteristics to generate a list of feasible treatment alternatives. The major question is how feasible alternatives can be identified. Three general approaches may be considered: (i) formulate the planning problem as a rule-based system to incorporate expert knowledge on feasible treatment trains; (ii) formulate the planning problem as mathematical optimization problem; and (iii) formulate the planning problem according to problem solving theory as described and used in Artificial Intelligence (AI). The drawback of the first approach lies in the additional requirement of expert knowledge. Furthermore the representation of such knowledge is not easily implemented. The result would be a rule-based expert system or to some extent a case-based reasoning system. In both cases solutions can only be derived if rules exist or similar cases are present within a knowledge base. This is contrary to the objective to achieve a most generic approach for flow sheet generation. The second approach, mathematical optimization, has been successfully applied to similar applications. Nevertheless, the requirements of a strict mathematical formulation as well as the provision of a superstructure containing all possible flow sheets prohibit the use within the scope of this work. The third approach addresses a wide field of different methods which all focus on a problem formulation based on the problem structure itself. A particular candidate of this area is the means-end analysis. Thereby repeatably subgoals are defined for the selection of possible treatment

units until an overall goal is achieved. In this case the final goal is the non-violation of effluent standards. All approaches of this field take the form of a search problem, or more advanced of a planning problem. The advantage is that the finding of flow sheets is domain independent. However these approaches are troubled by the phenomena of combinational explosion. This is because the problem space which contains feasible and infeasible solutions is of infinite size. The prior objective of the latter approach is to restrict the problem space to retrieve complete plans within a sufficiently short amount of time and calculation effort. In this work a search-based planning algorithm will be implemented to ensure a most generic implementation.

2
Literature Review

2.1 Introduction

As introduced in section 1.3 (Problem analysis) this work deals with (i) model development and (ii) model application for the purpose of flowsheet generation. The model itself captures knowledge on parameters for wastewater characterization (i.e. material model) and knowledge on options for treatment of wastewater (i.e. process model).

Regarding the material model, the review in section 2.2 evaluates the following questions. What parameter exist to characterize the qualitative state of a wastewater stream? What systemization approaches exist to order parameter for wastewater characterization? Particular interest lies in the identification of interactions between parameter regarding their assertions.

Regarding the process model, the review in section 2.3 evaluates the following questions. What processes can be applied to treat particular wastewater compounds? What phenomenological processes are included in which treatment technology? Which model approaches do exist to describe particular phenomenological processes? The findings of these sections must enable the process model to identify appropriate treatment technologies based on present wastewater compounds and furthermore to estimate the treatment efficiency based on the respective wastewater characterization.

The sections on wastewater characterization (2.2) and treatment options (2.3) resemble to the knowledge that is eventually represented by means of knowledge representation. Section 2.4 evaluates different forms and languages used for knowledge representation.

In this context, languages for knowledge representation belong to the model framework as introduced in figure 1.1. Regarding the model framework, the review in section 2.5 (Process Modeling) evaluates the following questions. Which approaches exist to organize modeling objects within a model framework? What approaches do exist to relate parameter to appropriate modeling approaches?

The above listed questions deal with issues related to the model development or in a broader sense with the development of a model framework. The second main focus lies on model application or from an AI perspective on knowledge processing. In particular the model application is about finding feasible treatment flowsheets. The review in section 2.6 (Process Design and Optimization) evaluates existing approaches for this task. The questions related to these issues are as follows. What planning approaches exist to identify feasible flowsheets from a given knowledge base on treatment options and given boundary conditions (i.e. given sources and sinks). How should one define the problem statement with regard to the overall objective. A desired feature of an appropriate planning approach is to achieve a most generic algorithm that does not require any further design rules or expert knowledge.

2.2 Wastewater characterization

2.2.1 Introduction

A wastewater stream is described by contained substances (dissolved or particulate) and properties that describe the overall wastewater state (e.g. temperature, pH value[1] or pressure). In the scope of this work properties of this nature are referred to as qualitative parameter and context properties.
In contrast quantitative parameter refer to the specification of flow of wastewater over time (e.g. flow per hour). Since the overall objective is the design of flow-sheets based on steady-state only constant flow has been accounted for in this work.
Within the context of process modeling, wastewater compounds are commonly referred to as state variables as long as the objective is to describe the fate of these variables over time. However aiming at a sound classification system of tangible wastewater constituents and pollutants the term qualitative parameter will be used instead of state variables.
In this context, qualitative parameter represent the result of commonly applied technical measurement methods.
Qualitative parameter represent direct or indirect chemical elements, or compounds within a particular wastewater. The form of appearance of a single chemical element or component depends on the surrounding context such as pH value, temperature or other chemical elements present.
The fundamental problem in characterizing the qualitative nature of wastewater is that compounds can in most cases not directly be detected. Either because

[1] In some model approaches pH value may also be represented by the presence of particular ions.

there are far too many compounds present (especially organic compounds) or the indirect detection of physical properties is much cheaper and efficient (e.g. measurement of Chemical Oxygen Demand (COD)).

Finally properties of compounds describe the form of appearance of chemical elements and components. Through properties a particular compound may be characterized as being dissolved or particulate, biodegradable or inert.

The latter relations hold also for chemical, biological or physical processes (or reactions). In short, phenomenological processes are characterized by its products/educts and process rates are dependent on the surrounding context (e.g. pressure, presence of oxygen and temperature).

As introduced above, the characterization of a wastewater stream can be achieved through the concepts of quality parameter, compounds, and properties of compounds, see figure 2.1. The three concepts are strongly related to each other. The introduction of these three perspectives are an anticipation of a later classification of wastewater characteristics. These three perspectives function as root concepts in an ontology on wastewater characteristics (see section 4.5).

Qualitative parameters are used for legal legislations (e.g. effluent standards) since they are based on standardized measurement techniques. At the same time qualitative parameter do either represent a particular chemical element or compound (e.g. ammonia) or they summarize wastewater constituents by a particular property (e.g. particulate matter measured as Total Suspended Solids (TSS)).

The concept of compounds is based on chemical classification only, regardless if a particular element or compound can be analytically detected or not (e.g. the substance ethanol may be present in a wastewater stream but it may not be measured in particular but may only be captured by the measurement of COD).

As a third concept properties of compounds play an important role regarding wastewater characterization on the one hand and regarding treatment of wastewater on the other.

The following two sections discuss important quality parameters and properties of compounds. The section on quality parameter also includes some information on analysis of particular chemical elements and compounds.

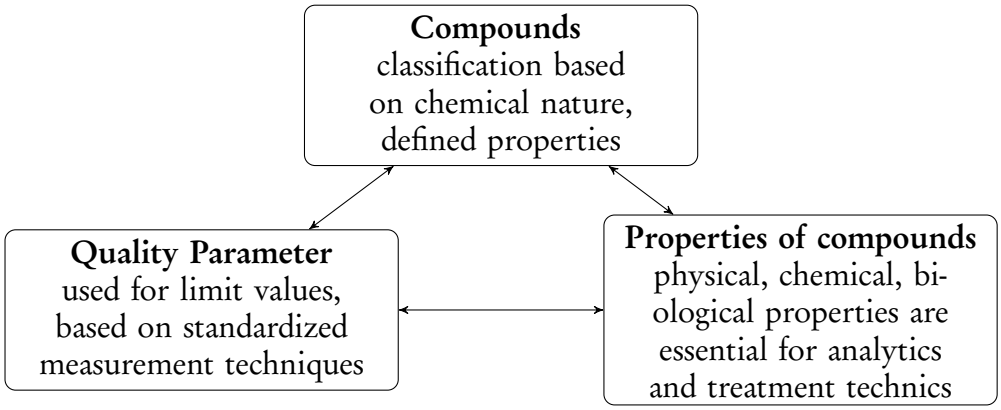


Figure 2.1: Systematization of wastewater characteristics though the perspective of parameter, properties, and compounds

2.2.2 Qualitative Parameters

2.2.2.1 Introduction

The objective of this section is to give an overview on classifications of common used qualitative parameters and to evaluate their assertions regarding overlapping content, representation of compounds and dependency on context properties. The findings of this review are then used in the development of a declarative material model for wastewater characterization (see section 4.5).

Following Koppe and Stozek (1993) one can differentiate qualitative parameters as: single substances (or even chemical elements), composite parameters, group parameters, and solids parameters (see also Schwister, K. ed. 2003).

Although the overall objective of this work lies on a generic approach to generate flowsheets for industrial wastewater most work on wastewater characterization is based on municipal wastewater.

Single substance parameters (or chemical elements) provide a maximum on information since their properties may be derived immediate by known context (e.g. a known concentration of ammonia and pH values allows immediate conclusion of concentration ammonium, see figure 2.2).

Composite parameters can further be differentiated into: (i) composite parameters derived by physical separation (e.g. by settling through gravity) such as Settleable Solids; (ii) composite parameters that represent compounds where a particular chemical element is present (e.g. Nitrogen, Carbon, Phosphor). In this case substances with different chemical bindings are covered. Nevertheless the end product of the measurement analysis yields the parameters of interest (e.g. organic carbon); (iii) composite parameters that cover compounds that show similar chemical behavior (e.g. alkalinity and acid capacity). An

Table 2.1: Systematization of quality parameters according to Koppe and Stozek (1993)

Type	**Definition**
single substance parameter	represent a particular chemical element or compound (e.g. NH_4)
composite parameter	summarizes a group of wastewater constituents based on a particular property (e.g. COD, N_{total}); further categorized as: • derived by physical separation • derived according to particular chemical element • derived according to similar chemical behavior
group parameter	summarizes wastewater constituents of similar chemical kind (e.g. water hardness)
solids parameter	summarizes wastewater constitutions based on physical properties such as particle size (e.g. TSS, VSS)

overview of the systematization is shown in table 2.1. An overview of compound parameters and a possible systematization is given in table 2.2.
Group parameters summarize chemical similar substances and compounds. An important an-organic group parameter is the degree of water hardness (Schwister, K. ed., 2003, p. 238). In the context of organic water constitutes there are the important groups of tenside, phenol, PAK etc.
Last but not least solids parameters summarize water constitutes according to physical properties such as density or particle size. Solids parameters do not differentiate according to chemical composition of compounds.

Table 2.2: Characterization of compound parameters in municipal wastewater (Koppe and Stozek, 1993, p.34-50)

Group	Parameter	Assertion
compound parameter for organic compounds	**glowing loss** filter dry residue - glowing loss	concentration of organic substances
	organic bound carbon total organic bound carbon - TOC dissolved organic bound carbon - DOC	concentration C
	oxidizability Chemical Oxygen Demand COD Biochemical Oxygen Demand BOD oxidizability with potassium permanganate (inorganic and organic compounds are detected)	concentration O_2 concentration O_2 concentration $KMnO_4$ (conversion to concentration of O_2 possible)
	organic bound nitrogen N_{org}	concentration N_{org}
	organic halogen compounds adsorbable organic halogen compounds (AOX) extractable organic halogens with Pentane, Hexan oder Heptan (EOX)	concentration AOX Konzentration EOX
physical and physicochemical parameter	coloring	
	turbidity	
	pH value	
	electrical conductivity	
	absorption of UV-radiation	
	redox voltage	
	density	
compound parameter for inorganic compounds	**total inorganic bound carbon (TIC)** prerequisite to measure TOC (TOC=TC-TOC or TOC=TC-TIC)	concentration C
	fixed residue SS - loss on ignition= dry matter - loss on ignition	concentration of mineral substances

Table 2.2 – continued from previous page

Group	Parameter	Assertion
	electrical conductivity κ detected are compounds which decompose into ions in aqueous solution (electrolytic dissociation)	$\mu S/cm$ or mS/m
other parameter (organic and inorganic compounds)	**total dry - and filtrate dry residue**	concentration
	suspended solids (SS) determine particulate compounds	concentration
	settlable solids all particulate settlable solids	volume portion on settlable solids
	mass portion of settlable solids gravimetric determination of settlable solids	concentration of SS
	toxicity toxicity to bacteria fish toxicity	percent LD50

2.2.2.2 Organic bound carbon

Especially wastewater of food industry is contaminated by organic compounds. The following parameters are used to describe these compounds.

Chemical Oxygen Demand

expressed as COD or COD_{total} (representing concentration of O_2). COD describes the volume related mass on oxygen required for the oxidation with potassium dichromate. Thereby one mol $K_2Cr_2O_7 \mathrel{\widehat{=}}$ relates to 1.5 mol O_2. Applied analysis method is described in DIN 38409 part 41.

Biological Oxygen Demand in 5 days

expressed as BOD_5 (concentration O_2). BOD_5 represents the mass concentration of dissolved oxygen required by respiration of microorganisms in 5 days. For analysis method see DIN 38409 part 51 (i) detect the change of oxygen concentration (ii) using an oxygen probe or by manometric approach.

2.2.2.3 Nitrogen

Nitrogen is encountered in different forms and is mainly described by the following parameters (see DIN 38405). An overview on the relations between nitrogen compounds is also given in table 2.3.

Table 2.3: Relations between nitrogen compounds

Form of nitrogen	**Abbrev.**	**Definition**
ammonia	NH_3	NH_3
ammonium	NH_4^+	NH_4^+
total ammonia nitrogen	TAN	$NH_3 + NH_4^+$
nitrite	NO_2^-	NO_2^-
nitrate	NO_3^-	NO_3^-
total inorganic nitrogen	TIN	$NH_3 + NH_4^+ + NO_3^- + NO_2^-$
total kjeldahl nitrogen	TKN	$N_{org} + NH_3 + NH_4^+$
organic nitrogen	N_{org}	$TKN - (NH_3 + NH_4^+)$
total nitrogen	TN	$N_{org} + NH_3 + NH_4^+ + NO_3^- + NO_2^-$

Total Nitrogen

expressed as N_{total} (representing concentration of N_{total}). Applied analysis method: reduction with Devarda's alloy[2] and catalytic decomposition, capturing also nitrite and nitrate (DIN 38409 part 28 1992). However this approach is hardly used. Ammonium, nitrite and many organic bound nitrogen compounds are heated with Peroxodisulfate in alkaline environment and high pressure. Thereby the nitrogen compounds are oxidized to nitrate and subsequently analyzed (DIN EN ISO 11905 (U36) 1998). Determination of bound nitrogen compounds TN_b - oxidization of all nitrogen compounds through catalytic incineration within an oxygen atmosphere at 700 °C to nitrogen oxides. Quantification on nitrogen concentration is achieved through Chemiluminescent detection (DIN 12260 (H34) 2003).

Organic Nitrogen

expressed as N_{org} (representing concentration of N_{org}). N_{org} is bound on biomass and compounds of organic origin (e.g. protein). Applied analysis method: deduction as Kjeldahl Nitrogen with further additives and measured as ammonia ($TKN - (NH_3 + NH_4^+)$).

Ammonium Nitrogen

expressed as $NH_4 - N$ (representing concentration of $NH_4 - N$). Ammonium (NH_4^+) and ammonia (NH_3) dissociate into each other depending on pH value (Figure 2.2). Within neutral or acid milieu ammonium ions prevail. Within alkaline milieu ammonia ions prevail in dilution. Applied analysis method: Ions are transferred through reagents into indophenol blue and are determined spectrophotometrically at a wave length of 690 nm.

[2] An alloy of aluminum, copper, and zinc (in the approximate ratio of 45%, 50%, 5%).

Nitrate Nitrogen

expressed as $NO_3 - N$ (representing concentration of $NO_3 - N$). Nitrate is the end product of the process of nitrification. Applied analysis method: through evaporation nitrate evolves to sodium salicylate. After dissolution with sulfuric acid the reactants are transferred into a yellow dye which is measured spectrophotometrically. In particular does nitrate react within sulfuric acid with 2,6-dimethylphenol to 4-nitro-2,6-dimethylphenol which is measures spectrophotometrically at a wave length of 324 nm (DIN 38405(9)).

Nitrite Nitrogen

expressed as NO_2-N (representing concentration NO_2-N). Nitrite occurs in wastewater as effect of incomplete nitrification of ammonium. Applied analysis method: It shows a slight organic color reaction and reacts in an acidic solution with sulfanilamide, which then by coupling with N-(1-naphthyl)-ethylenediamine is a red dye. Subsequently the amount is measured spectrophotometrically at a wave length of 540 nm.

Total Kjeldahl Nitrogen

expressed as TKN (representing concentration of TKN). TKN captures most nitrogen compounds with oxidization number 3 (trivalent negative nitrogen). Applied analysis method: within the conventional approach a defined sample volume is broken down with sulfuric acid in a Kjeldahl Bulb. The organic parts of the sample are removed and the nitrogen is converted into ammonium sulfate. With the addition of a strong base, ammonia is released from the digestion solution, which is collected in an acid, and is determined by titration. Due to the fact that nitrogen compounds are easily decomposed usually only TKN and N_{total} are applied for wastewater characterization.

2.2.2.4 Phosphorus

Phosphorus occurs in aquatic systems mainly in the form of orthophosphate and organic bound phosphorus. The following parameters describe occurrence of phosphorus in the context of wastewater characterization (Geiger and Nafo, 1999).

- **Total Phosphorus** expressed as P_{total} (representing concentration of P_{total})

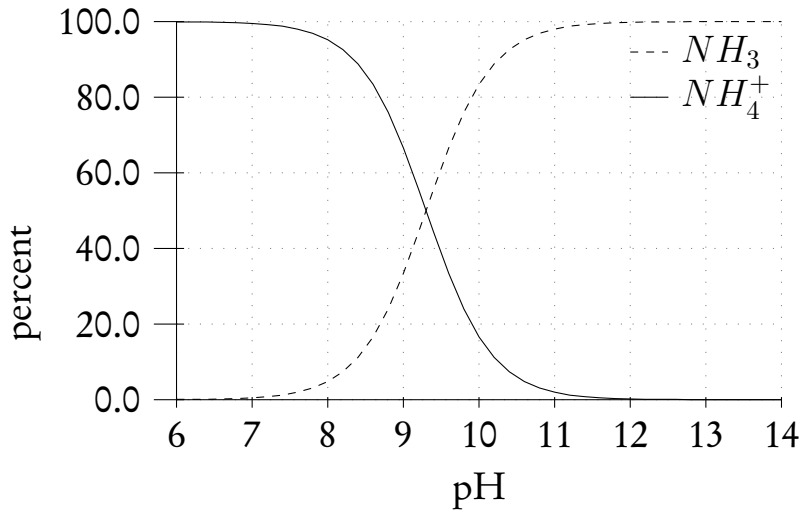


Figure 2.2: Equilibrium $NH_4^+ \rightleftharpoons NH_3 + H^+$ ($pKs = 9.25$)

- **Phosphate or Orthophosphate** expressed as PO_4 or $o - P$ (representing concentration of PO_4). Is usually fully oxidized and is furthermore required for biological growth.

- **organic bound phosphorus** expressed as $org - P$ or P_{org} (representing concentration of P_{org})

Usually only total phosphorus and orthophosphate are measured. For untreated wastewater the difference between the two parameters can be attributed to organic phosphorous.

2.2.2.5 Fats and oils

Fats and related compounds are summarized under the term lipids (Koppe and Stozek, 1993, p. 62). An important parameter of this compound groups is **low-volatility lipids** which refers to un-polar compounds with $K_p \geq 250$ °C. Applied analysis method: through extraction with petroleum-ether or hydrocarbons.
Lipophilic substances tend to be less bioavailable due to their hydrophobic character and high melting point (Reimann *et al.*, 2002). Due to their chemical structure fats are also detected by qualitative parameter for organic carbon such as COD.

2.2.2.6 Dissolved, suspended and particulate matter - Solids Parameters

Besides wastewater characterization based on chemical compounds commonly physical properties such as particle size play an important role in wastewater

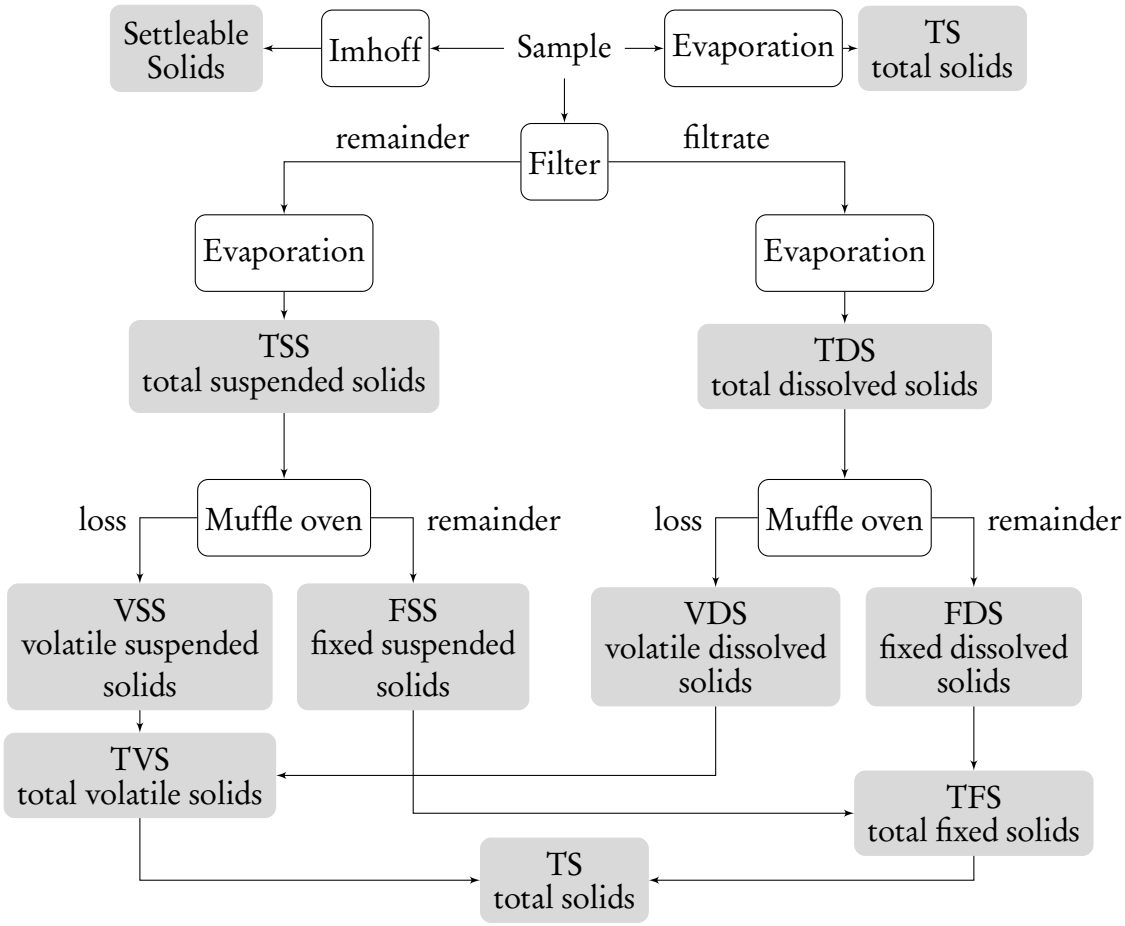


Figure 2.3: Solids parameters

characterization. Usually water constituents are described as dissolved or particular matter. The following parameters are used to describe solids parameters. The description on analysis methods for solids parameters is given in DIN 38409.

Total Solids TS (mg/l)

Measurement method: dry a defined probe at 105°C, the remainder is expressed as TS (dissolved salts are captured as well).

Total Suspended Solids (mg/l)

is the amount of particulate matter within a probe after filtration and drying.

Volatile suspended solids VSS (mg/l)

defines the organic amount of particular matter within a probe. Through incineration all organic and inorganic matter is oxidized.

Settleable Solids - SS (ml/l)

defines the volume amount of particulate matter that settles within a given amount of time (e.g. 30 Minutes).

Organic matter is usually bound to particulate matter and hence solids parameter are related to parameter such as BOD_5, COD as well as to organic nitrogen and phosphorus (Leinweber, 2002, p. 107).

An overview regarding the relation between solids parameter and the means for their derivation is shown in figure 2.3.

Besides the coarse differentiation used for solids parameters the analysis of present particle size distribution gives a much more detailed view on the distribution of wastewater constituents. Figure 2.4 shows important particle types and possible measurement techniques. In practice however a detailed analysis of particle size distribution is usually not performed.

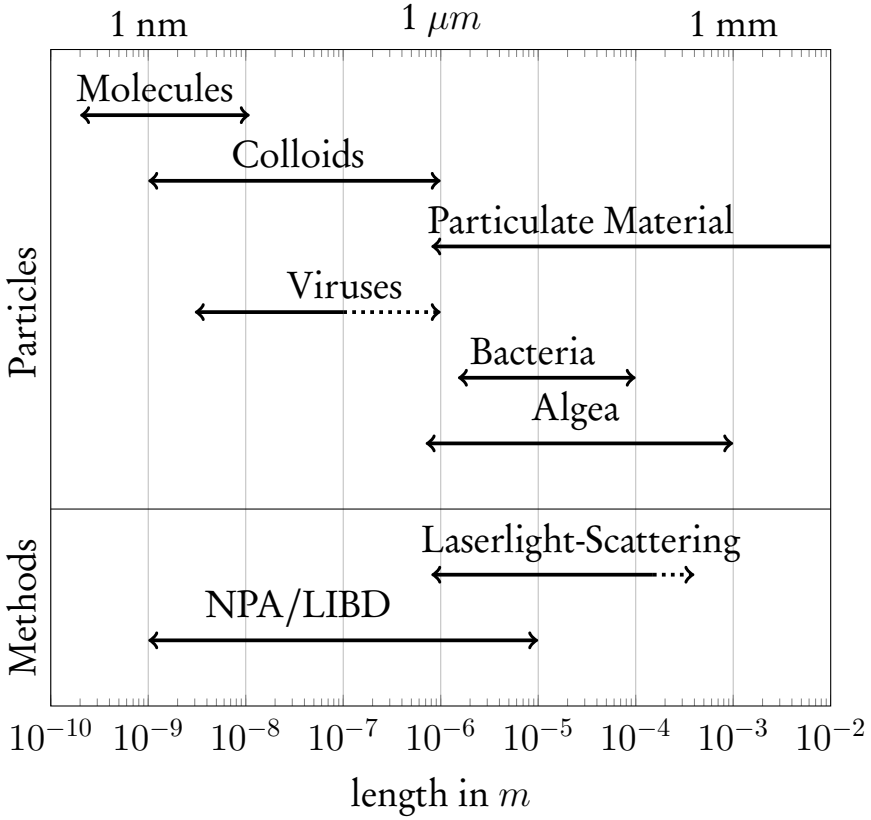


Figure 2.4: Spectrum of contaminants regarding particle size range and measurement techniques

2.2.3 Properties of compounds

2.2.3.1 Introduction

In the latter section important quality parameter and compounds have been presented. These parameter can be detected because of specific properties. In this section important physical and chemical properties are outlined.
It is important to note that wastewater is a heterogeneous mixture of different compounds wherein the properties of included substances do interrelate.

2.2.3.2 Physical properties

With respect to particulate wastewater constituents primarily particle size, particle size distribution as well as particle density are of primary interest.
As context property temperature affects the solubility of constituents as well as other chemical and biological reactions. In particular increased temperature decreases the solubility of gases in water. The metabolic rate of aquatic organisms is also related to temperature.

Particle size (dispersion)

Any discritization regarding particle size distributions of present water constituents is a simplification of reality. For a very practical approach a simple distinction between dissolved and particulate compounds is usually made. Thereby this distinction depends on the pore size of the filter used (e.g. 4 μm). A more fine grained distinction regarding particle size is given in table 2.4.

Particle density

Within gravimetric based separation processes particle density plays an immanent role. It is obviously important that the density of the particle to be separated must be higher than the one of the surrounding medium (or lighter in case of flotation techniques). The density of the surrounding medium can be influenced through the concentration of dissolved salts or other constituents.

Table 2.4: Grain classes (Rickert and Hunter, 1971)

Class	**Range**
settleable	$> 100\ \mu m$
supra-colloidal	$1...100\ \mu m$
colloidal	$1\ nm...1\ \mu m$
soluble	$< 1\ nm$

Furthermore, particles with a large particle size tend to settle better than particles with smaller particle size which is described in Stokes law (see equation 2.5).

2.2.3.3 Chemical properties

The properties of compounds in wastewater are dominated by the surrounding medium water. Thereby water molecules can be understood as polar solvents. Therefore water takes part in many reactions with wastewater constituents. The most important two are hydrolysis and hydration (Koppe and Stozek, 1993, p. 23-24). The process of hydrolysis leads to splitting up of complex organic compounds into bio-available organic matter. Within hydration water molecules are attached to dissolved ions, atoms, molecules or colloids. Thereby the H_2O bound remains unchanged.
Through water as a polar solvent constituents become dissolved. Hydrophilic groups within organic compounds as for example $-OH$, $-COOH$, $-NH_2$ and $-CHNH_2$ make these compounds polar and hence increase solubility . Beyond this salts show the highest solubility in water. Dissolving of compounds in water leads to a change of viscosity, surface tension, overall density and acid-base capacitance (buffer capacity).
Beyond the above mentioned chemical properties other relations between water and water constituents exist which are not further elaborated in this context.

2.2.3.4 Relation between bio-degradability and particle size

Next to single properties of wastewater constituents also interrelations between compound properties exist. An important interrelation is the one between particle size and biodegradability.
Regarding particle size distribution of wastewater, research results have been published in the following articles: Delekgurgen *et al.* (2006); Levine *et al.* (1991); Sophonsiri and Morgenroth (2004); Vaillant *et al.* (1999). An essential finding of this work is that particle size distribution is particular to each wastewater. Therefore the particle size distribution of a wastewater has been referred to as finger print of a particular wastewater by Delekgurgen *et al.* (2006).
Nevertheless some findings can be generalized. Organic material must have a minimal particle size before it is readily biodegradable. The breakup of complex compounds into readily biodegradable compounds is accomplished by the process of hydrolysis. Some examples are given in table 2.5.

Table 2.5: Relation between bio-degradability and dispersity on examples (Koppe and Stozek, 1993, p. 53)

degradability	dispersity	example
fast bio-degradable	mostly dissolved	acetic acid, ethanol
slow bio-degradable	partial dissolved	emulsified fat, starch
almost not bio-degradable	dissolved and particulate	EDTA, cellulose

In the work of Sophonsiri and Morgenroth (2004) the relation between wastewater load and particle size distribution was investigated. Key quality parameter in focus was COD and P_{total}. Organic compounds have been further classified into protein and carbohydrates. The achieved treatment efficiency has been investigated for three different types of wastewater as shown in figure 2.5.
In the result of the work of Sophonsiri and Morgenroth (2004) as shown in figure 2.5 it can be seen that the three types of analyzed wastewater show a high load of contaminants attached to particles of large size. The different treatment technologies applied primarily have the effect to cut peaks related to the high concentrations bound to large particles.

2.3 Treatment options and modeling approaches

2.3.1 Introduction

A large number of different technologies exist for treatment of municipal and industrial wastewater. Even more, most approaches allow for various possible technological implementations. For this reason this section will focus only on selected physical (mechanical) and biological treatment approaches. In contrast chemical approaches are neglected as they have not been considered in the context of this work.
The main questions that are investigated in this section regarding the overall objective of this work are:

- What phenomenological processes are applied to treat particular wastewater compounds?
- What phenomenological processes are included in which treatment technology?
- Which model approaches do exist to describe particular phenomenological processes?

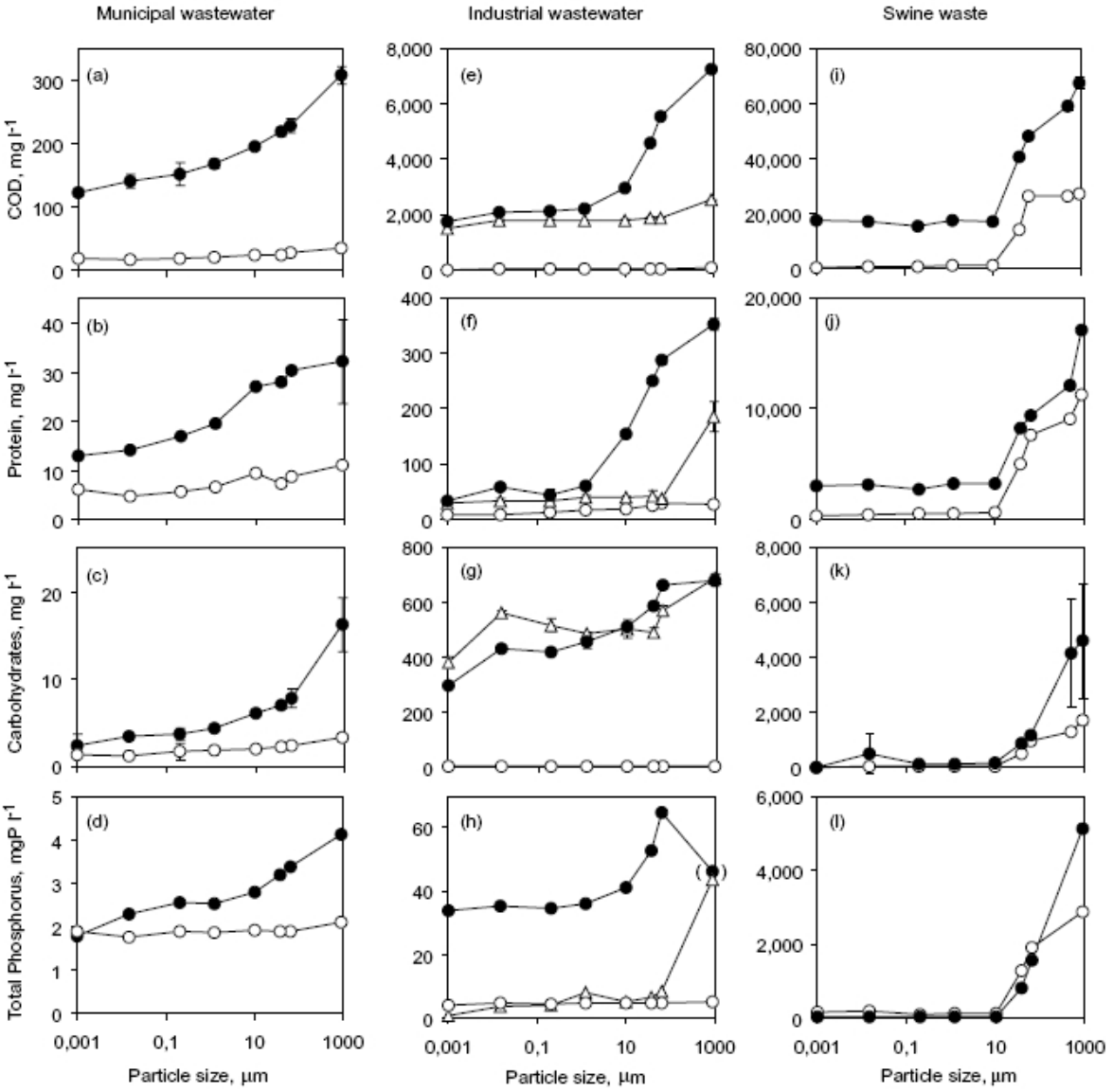


Figure 2.5: Comparison between Particle Size Distribution of municipal wastwater (•= effluent primary treatment; ○= effluent secondary settler) industrial wastewater (•= buffer tank; △= effluent flotation, ○= effluent secondary settler) and wastewater from a pig farm (•= effluent pigsty; ○= effluent wastewater pond) published in Sophonsiri and Morgenroth (2004)

These questions relate to the overall purpose of the approach to establish a knowledge base on treatment units that describe the treatment efficiency based on given wastewater characteristics, see figure 2.6.

From a modeling point of view it is thereby important to make a distinction between phenomenological processes and treatment technologies. This results from the objective to achieve a most generic process description to allow for an application of different types of wastewater. This in turn requires a clear definition of the relation between particular wastewater constituents and the performance of a treatment process.

For a qualitative description of such relations see table 2.6 and figure 2.7. In contrast numerous process descriptions exist based on types of wastewater (e.g. estimated treatment efficiency on COD through flotation for pulp and paper industry wastewater). However such wastewater specific definitions of treatment efficiency are not generic and hence are not useful in the context of this work.
For the purpose to describe treatment options (or treatment units) as unit operations within a flowsheet based model representation (see section 2.5.7.3) the description must include each of the following:

- process background and included materials (educts and products)
- available modeling approaches and design formalism (preferable as 0-d models to represent process units)
- context and boundary conditions (pre-conditions)
- cost information of possible technological implementations (e.g. running cost as energy consumption per cubic meter and investment cost.)
- dimensioning algorithm of possible technologies (e.g. size of reactor volume)

As a first step in organizing treatment processes all processes will be characterized as either being a conversion process or a separation process.
According to Gujer (2008, p. 70) "in transformation processes materials are converted from educts (raw materials) to products by either chemical reactions or processes catalyzed by living organisms, in particular microorganisms. We characterize such processes by defining state variables (material concentrations), stoichiometry (relationships of educts and products) and kinetics (rate of the process)."
The change over time of mass of a particular substance dm/dt results from the difference from the inflowing and outflowing mass within the observed

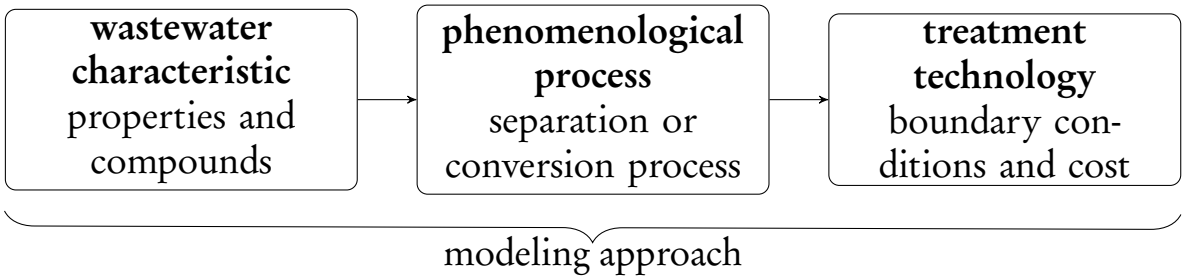


Figure 2.6: Causal chain of wastewater characteristics, applied phenomenological treatment processes and implemented technology

Table 2.6: Treatment technologies with respect to particle size (Imhoff and Imhoff, 1993, p. 89)

Particle characteristic	**Technology**
coarse suspended matter	sieve and rake
coarse settlable matter	grit chamber
oil and grease, colloids	oil trap, flotation tank, settling tank with skimmer
fine suspended matter	settling tank, flotation tank, chemical precipitation, micro-sieve, sand filtration
dissolved, partly-dissolved and fine organic matter	biological treatment

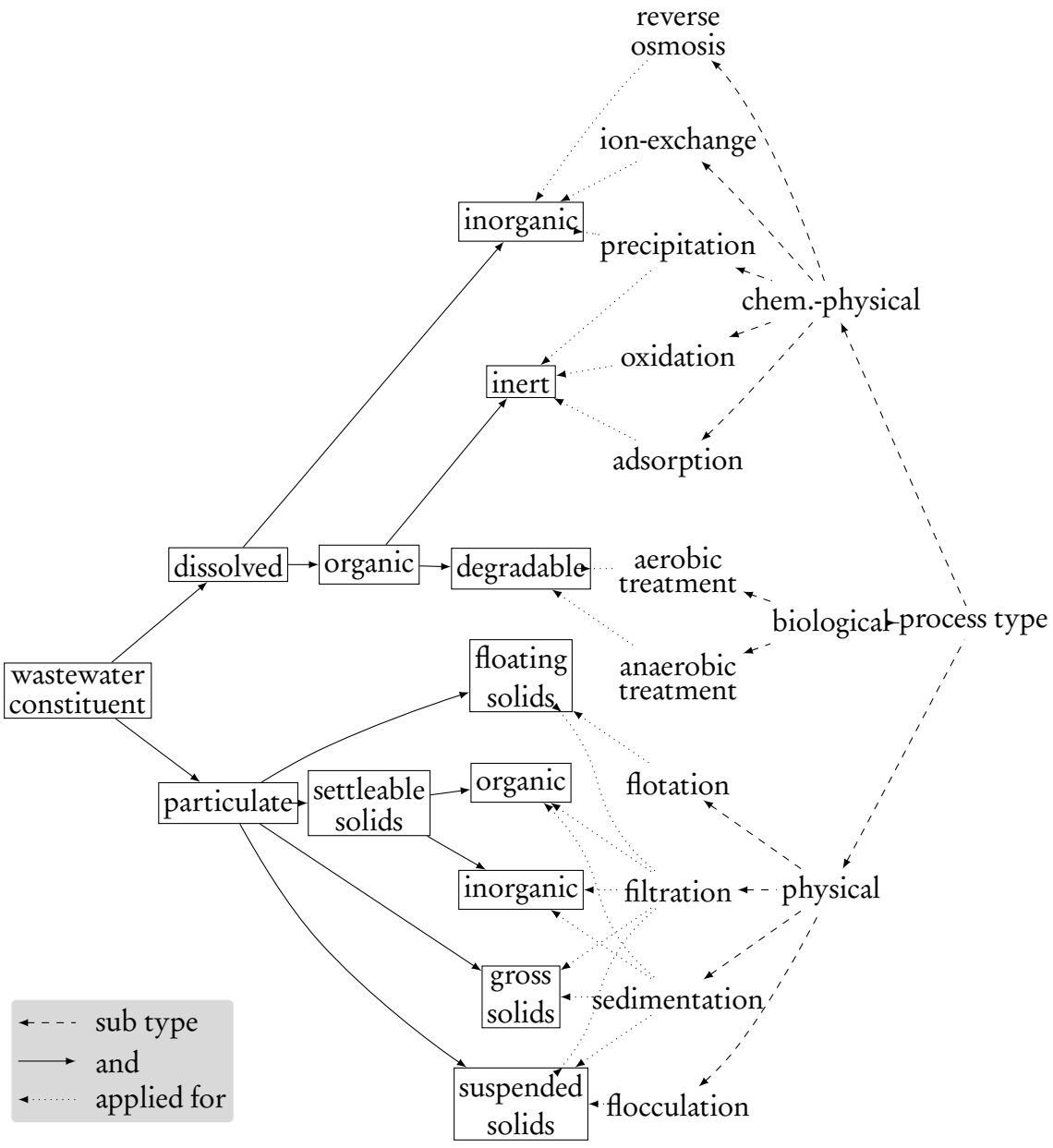


Figure 2.7: Relation between wastewater constituents and treatment processes

compartment (Gujer, 2008), see equation 2.1. Commonly such a compartment is assumed to be modeled as a Continuously Stirred Tank Reactor (CSTR). For convenience the system may be described in terms of concentration as shown in equation 2.2. In equation 2.2 L_s represents the inflowing load in kg/s, V is the volume of the compartment in m^3, C is the concentration within the compartment in kg/m^3, k is an outflow rate in 1/s, and r describes a possible change of substance concentration due to a conversion process.

$$\frac{dm}{dt} = \dot{m}_{in} - \dot{m}_{out} + \dot{m}_{transformed} \tag{2.1}$$

$$\frac{dC}{dt} = \underbrace{\frac{L_s}{V} - k \cdot C}_{\text{input and output}} + r \tag{2.2}$$

For separation processes performed in reactors with two outlets, overflow and underflow, the mass balance of an observed substance concentration may be represented by (Dick and Young, 1972, p. 36):

$$Q_{in} \cdot X_{in} = Q_O \cdot X_O + Q_U \cdot X_U \tag{2.3}$$

With Q_{in}, Q_O, Q_U being the inflow, overflow and underflow of the separation facility and X_{in}, X_O, X_U being the concentration of the particulate matter of the inflow, overflow and underflow. In the equation above conversion processes are neglected. Given a elimination or separation efficiency of the concentration of the overflow the above equation can be rewritten as follows. Thereby E represents a constant efficiency. Equation 2.3 can now be rewritten as follows:

$$\begin{aligned} X_u &= \frac{Q_i X_i - Q_o X_o}{Q_u} \\ \text{with } X_o &= X_i - X_i \cdot E \\ X_u &= \frac{X_i(Q_i - Q_o + Q_o E)}{Q_u} \end{aligned} \tag{2.4}$$

The following two subsections give a more detailed view on physical and biological processes related to wastewater treatment.

2.3.2 Physical phenomena and application in treatment processes

2.3.2.1 Introduction

Physical phenomena are fundamental for mechanic treatment technologies in wastewater treatment. This is because municipal and industrial wastewaters comprise a multitude of different compounds present in different forms, which prohibits in most cases a compound specific treatment strategy. However all compounds which are not truly dissolved obey to the laws of gravity or may be addressed by their size. The following physical phenomena are related to mechanic treatment approaches: absorption, adhesion, adsorption, buoyancy, dispersion, aeration, filtration, flocculation, coagulation, sedimentation, matter transport, and turbulence.

Hunze (2005) defines these phenomena as follows. Absorption refers to the take up of gas by fluids or solids. Absorption results in an increase of volume of the phase mixture due to the even distribution of gas within the surrounding material (e.g. due to aeration). Adhesion describes the phenomenon of two obstacles sticking together driven by near field molecular attraction (e.g. used in flocculation). Adsorption is about the deposition of substances or particles upon a phase surface. The phenomenon results from mutual attraction between surfaces (van der Waals forces). An illustrative example is the attachment of activated sludge flocs to gas bubbles. Furthermore, any wastewater constituent dissolved or particulate may adhere to activated sludge flocs. The inverse phenomenon to adsorption is referred to as desorption.

Buoyancy results from different densities of particles and surrounding fluid. Due to a displacement of water from particles with a smaller density than the surrounding liquid those particles experience an upward force. The phenomenon is gravity based and is dependent on the density of the particles and surrounding phase.

Aeration describes the dissolution of oxygen within liquid phase from atmospheric oxygen. It takes place at the liquid surface and is based on the phenomenon of diffusion. Filtration allows for the separation of particulate or suspended substances from the liquid phase. Whilst the phase system is lead through a given filter material solids are hold back within the filter material. The fundamental phenomenon involved is adsorption in case of depth filtration. The phenomenon depends on particle size and material properties.

Flocculation describes the forming of large flocs from smaller ones. The phenomenon results from the following two processes:

1. Collision of two or more flocs induced by the the surrounding flow regime. The trajectories of each floc result from Brownian motion, downward sinking of a particles due to gravity and gradients of velocity.
2. Coagulation which results in forming connections between flocs.

The process of flocculation is influenced by physical, biological and chemical boundary conditions such as turbulence or the electrostatic charge of flocs. Flocculation is opposed by the process of floc breakage.
Sedimentation, similar to the phenomenon of buoyancy, is induced by gravity. Particles with a density higher than the one of the surrounding liquid sink downward. Major influences on sedimentation are size, density and structure of particles as well as the concentration of particles and the flow field. Matter transport is achieved either by flow (advective transport) or due to differences in concentration (diffusive transport).
Turbulence describes the uneven distribution of pressure and fluctuation of fluid parcels. The phenomenon brings about an intermixing of a fluid and contained constituents within a fluid body.
The problem of modeling physical phenomena applied for mechanic treatment lies in the difficulty to measure the present particles within a phase system in terms of particle size distribution and density distribution in space. Both particle size and particle density occur in large ranges within wastewater phase systems.

2.3.2.2 Sedimentation

Gujer (2008, p. 40) defines sedimentation as "particles sediment in water, if additional, outside forces affect the particles (gravitation, buoyancy, centrifugal forces, inertia). Thus, particles are diverted from the flowing water."
Sink velocity for spheric particles under laminar flow regime (Re < 1) can be derived according to Stoke's law by (see Gujer 2008, p. 40):

$$v_s = \frac{(\rho_p - \rho_f) \cdot g \cdot d^2}{18\eta} \tag{2.5}$$

Thereby v_s is the settling velocity in m/s, d_p particle diameter, η_1 dynamic viscosity of water (1.0034 mm^2/s at 20 °C), ρ_1 and ρ_p density of water and the particle and g the acceleration due to gravity. The above equation holds only for idealized spheric particles at low concentrations.

The difference between primary and secondary settling is that in the first interaction between particles can be neglected whereas in the second not (due to hindered settling)[3]. Hence for secondary settling Stoke's Law can not be applied for secondary settling (Walter, 2005). Hindered settling occurs for waste streams with a solids concentrations beyond 2 %.
The settling velocity for hindered settling may be described according to the work of Vesilind (1968). The Vesilind Equation describes the settling velocity of activated sludge in terms of the particle concentration X and two constants V_0, k.

$$v_s = v_0 \cdot exp^{-k \cdot X} \tag{2.6}$$

Different approaches besides the Vesilind equation for the description of the settling velocity for hindered settling can be found in Akça *et al.* (1993); Dochain and Vanrolleghem (2001); Grijspeerdt *et al.* (1995); Hunze (2005); Walter (2005).
The design criteria for settlings tanks is usually the active surface (A_S), as defined in the below equation (Walter, 2005) with Q_i the inflow and v_s a generalized settling velocity (equation 2.5).

$$A_S = \frac{Q_i}{v_s} \tag{2.7}$$

Decanter and centrifuge

Similar to sedimentation the effect of centrifuges arises from differences of density between particles and surrounding fluid. In contrast to sedimentation the driving force is not acceleration of gravity but acceleration due to centrifugal forces (Walter, 2005, p. 16). Based on Stoke's Law the sedimentation velocity can be calculated with the dimensionless factor C, see equation 2.8 and 2.9.

$$C = \frac{r_m \cdot \omega_{dr}^2}{g} \tag{2.8}$$

$$v_s = \frac{d_p^2(\rho_p - \rho_f) \cdot C}{18 \cdot \eta} \tag{2.9}$$

In the above equations r_m relates to the average radius in meter and ω_{dr} relates to the angle velocity in 1/s. An overview on the application range of decanter and centrifuges is given in figure 2.8

[3]Except for diluted sludge.

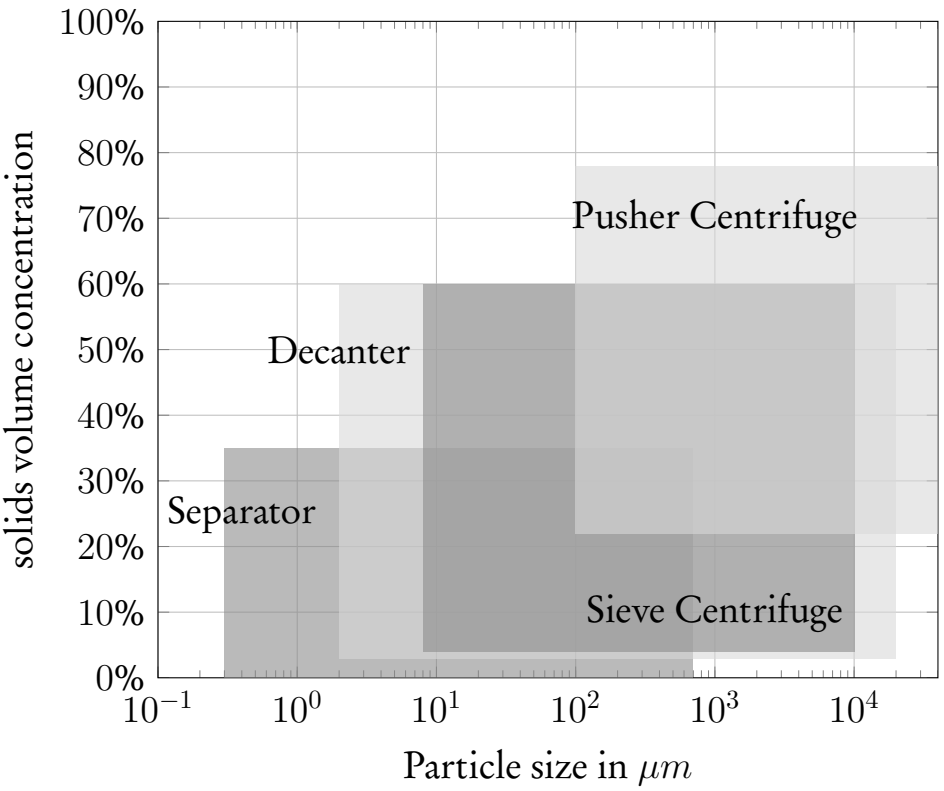


Figure 2.8: Application range centrifuge technology (Stahl, 2004)

Primary settler

The purpose of a primary settler is the separation of suspended solids. Overflow rate and influent solids concentration are the two important parameters on the efficiency of primary settling. In particular solid removal efficiency increases with increasing influent solids concentration and decreases with increasing overflow rate (Hunze, 2005).

Smith (1969) relates the removal efficiency of suspend solids of the influent (SS_i) according to equation 2.10. Tebbutt and Christoulas (1975) additionally include the influent concentration of suspended solids according to equation 2.11.

$$E = 0.82 \cdot e^{-0.0088 \cdot q} \tag{2.10}$$

$$E = 1.138 \cdot e^{-\left[\left(\frac{358}{SS_i}\right)+0.0016 \cdot q\right]} \tag{2.11}$$

In the above equations q relates to the surface overflow rate in $m^3 m^{-2} d^{-1}$. The predicted removal efficiency as described in equation 2.11 is also depicted in figure 2.9.

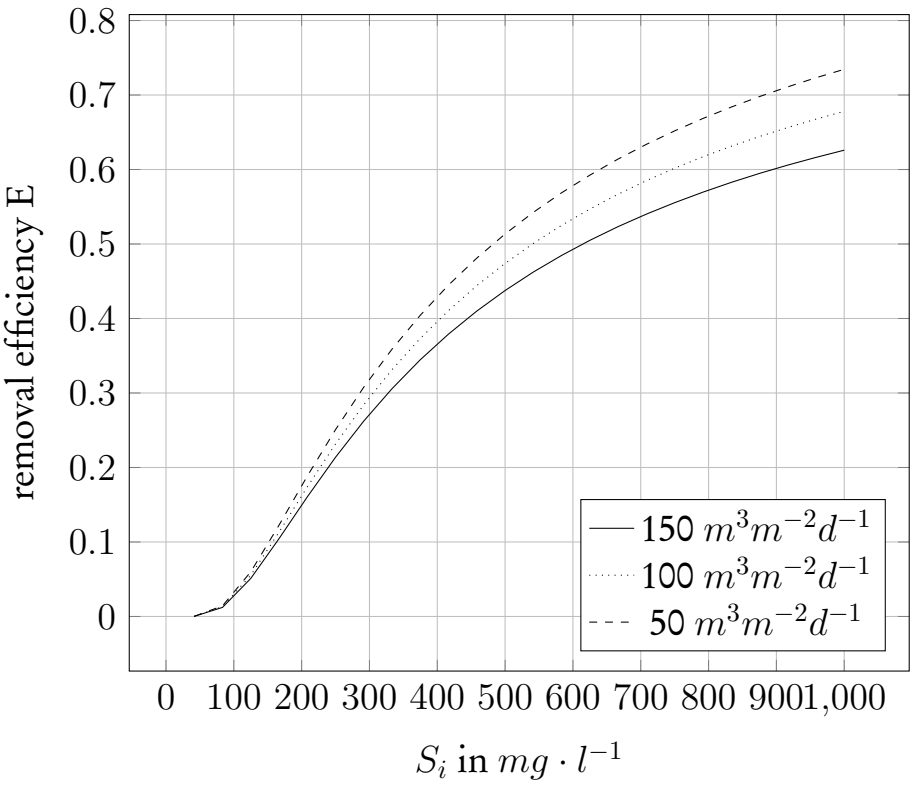


Figure 2.9: Effect of influent suspended solids on removal efficiency according to equation 2.11

Secondary settler

Lessard and Beck (1993) proposed to model secondary settlers by a clarification function and a thickening function. The clarification function assumes two zones within the settling tank. That are a clarification zone with a fixed volume and a below dead zone with variable volume. Through the clarification zone all conservative pollutants are routed to the overflow. The suspended solids concentration is determined by the relationship given in equation 2.12.

$$SS_O = a_1 + a_2(Q_F + Q_U) \tag{2.12}$$

Thereby relates SS_O to the suspended solids concentration in the overflow, Q_F and Q_U represent the inflow feed and the underflow, a_1 represents the minimum concentration in the effluent and a_2 is a proportionality constant for the effect of flow on effluent suspended solids. In the work of Lessard and Beck (1993) the parameter a_1 and a_2 have been estimated with 3 $g \cdot m^{-3}$ and 0.009.

The thickening function is based on a conventional flux theory and assumes a thickening zone as well as a below compression zone.

$$\frac{dVSS_{sb}}{dt} = \frac{(MLVSS \cdot (Q_F + Q_R) - VSS_O \cdot (Q_F - Q_W))}{A} - \frac{VSS_O \cdot (v_S + v_u)}{SBHT} \quad (2.13)$$

Thereby the concentration of VSS in the sludge blanket over time is a function of the concentration of the VSS concentration in the aeration tank (Mixed liquor volatile suspended solids (MLVSS)) and overflow (VSS_O), the Sluge Blanket Height (SBHT), the feed-, recycle-, and waste-flow, as well as the settling velocity of the clarification zone and the underflow. An alternative approach for modeling sludge thickening is given by Giokas *et al.* (2002).

A simple dynamic two compartment model for the representation of a secondary clarifier is described by Olsson and Newell (1999, p. 93-94). Here, the two compartments represent a clear water and a sludge compartment. The sludge hold up M_C within the clear water zone is given by equation 2.14. The sludge holdup within the sludge compartment M_S is represented by equation 2.15. Eventually the sludge concentration within the sludge compartment is calculated with equation 2.16.

$$\frac{dM_C}{dt} = \alpha q_F X_F - Q_O X_O - m_S \quad (2.14)$$

$$\frac{dM_S}{dt} = (1-\alpha) Q_F X_F - Q_U X_U + m_S \quad (2.15)$$

$$\frac{dX_S}{dt} = \left(1 + \frac{M_S}{K_T}\right)\frac{X_F}{r} - \frac{X_S}{r} \quad (2.16)$$

Equation 2.16 represents a thickening model which relates the concentration of the sludge blanket X_S to the concentration of the feed X_F and the sludge mass with a time lag. The sludge concentration of the over and under flow can be calculated with equations 2.17 and 2.18.

$$X_O = \frac{M_C}{V - V_S} \quad (2.17)$$

$$X_U = K_C X_S \quad (2.18)$$

Table 2.7: influencing factors on the effectiveness of flotation (Walter, 2005, p. 19)

Gaseous phase	Liquid phase	Solid phase
rigid influencing parameter		
water temperature and solubility of the gas	viscosity	concentration
size distribution of bubbles	surface tension	particle size- and density distribution
collision effectiveness	conductivity, pH value	surface properties
adjustable parameter		
bubble diameter	surface feed	solids feed
rising velocity	recycle flow	thickening time
saturation pressure		
realization of the relaxation process		

In equation 2.18 K_C is a constant of proportionality. Different design approaches for secondary settlers such as STOWA, ATV, and WRC design are explained in Henze *et al.* (2008, p. 327-329).

2.3.2.3 Flotation

The concept of flotation is based on equal physical laws as sedimentation. In contrast, due to raising air bubbles pull attached particles to the fluid surface (phenomenon of adsorption) (Walter, 2005, p. 18).
The uprising velocity of air bubbles is between 0.01 to 0.1 m/s. The smaller the diameter of air bubbles the easier particles become attached to them. However also the uprising velocity decreases with smaller bubble diameter.
So far no reliable models for the description of the flotation process exist. However a qualitative overview on influencing parameter regarding flotation is given in table 2.7.
In contrast to the above listed separation technologies membrane technology allows for separation of much smaller particles as shown in figure 2.10.

2.3.3 Biological Processes

2.3.3.1 Phenomenological processes

Biological processes play an important role for the treatment of wastewater both in municipal and industrial wastewater. The heart of the process is copied from processes as they occur in natural rivers where micro organisms oxidize carbon sources present the water. Those processes are in general limited by the scarcity of carbon sources and dissolved oxygen. Within technical applications boundary conditions are optimized as such to enable the efficient treatment of

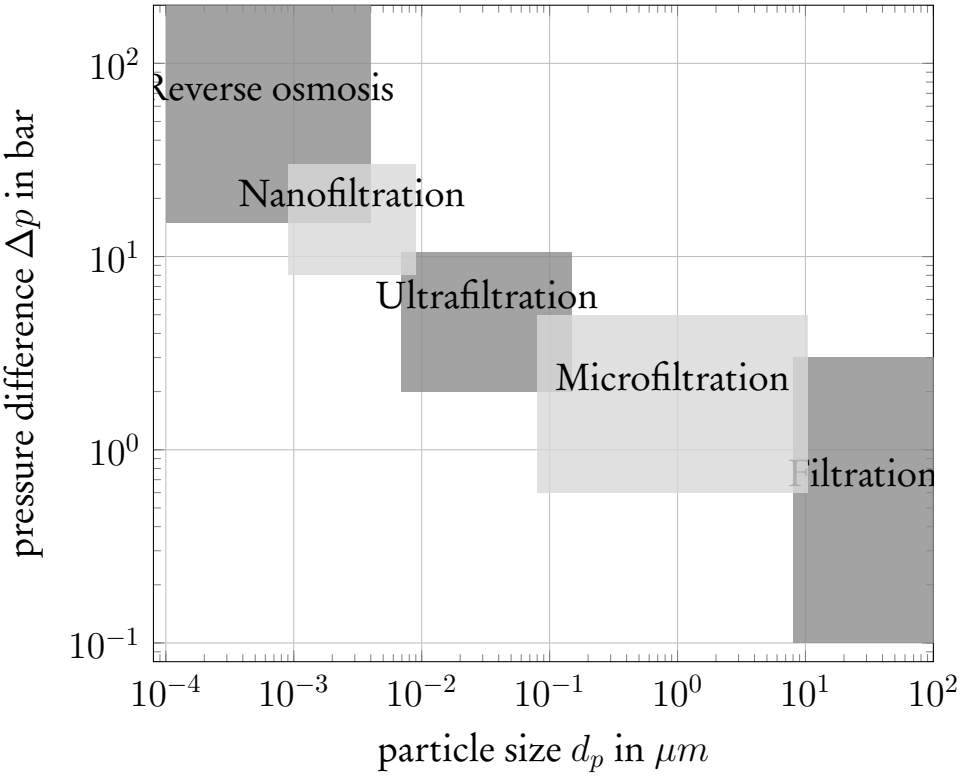


Figure 2.10: Application range of pressurized membrane technology (Rautenbach, 1997)

pollutants. The description of biological processes is based upon the work of Henze *et al.* (2008); Hunze (2005); Walter (2005).

Oxidation of organic carbon compounds

relates to the respiration of organisms during which biodegradable carbon sources are oxidized as energy source as well as to build up biomass. Therefore the organic carbon compounds must be readily degradable (converted from particulate to dissolved compounds through hydrolysis. Heterotrophic biomass subsequently uses organic compounds through oxidation as energy source. Thereby, oxygen takes the role of an electron acceptor. Oxygen may be present in dissolved or bound (see denitrification) way.

$$carbon + O_2 \xrightarrow{\text{microorganism}} biomass + CO_2 + H_2O \tag{2.19}$$

Denitrification

is about the reduction of oxidized nitrogen compounds resulting in nitrogen gas. Many heterotrophic microorganisms are able to change to nitrogen-respiration. However the exertion of nitrogen-respiration is only possible through

the absence of dissolved oxygen (Akça *et al.*, 1993; Sykes, 2003).

$$NO_3^- + 2H^+ + 2e^- \rightarrow NO_2^- + H_2O \quad (2.20)$$
$$NO_2^- + 2H^+ + e^- \rightarrow NO + H_2O \quad (2.21)$$
$$2NO + 2H^+ + 2e^- \rightarrow N_2O + H_2O \quad (2.22)$$
$$N_2O + 2H^+ + 2e^- \rightarrow N_2 + H_2O \quad (2.23)$$

Oxidation of nitrogen compounds (i.e. Nitrification)

is about the oxidation of ammonium to nitrate. Autotrophic micro organisms do not require organic carbon sources instead they use carbon dioxide as carbon source. Compared to heterotrophic organisms slower growth rates occur which in turn require longer sludge ages to ensure sustainable growth of such organisms.

$$\text{nitritation } NH_4^+ + 1.5O_2 \rightarrow NO_2^- + H_2O + 2H^+ \quad (2.24)$$
$$\text{nitration } NO_2^- + 0.5O_2 \rightarrow NO_3^- \quad (2.25)$$

Not mentioned in this context are the processes of biological phosphorous elimination as well as anaerobic processes.

2.3.3.2 Modeling of biological removal of carbon and nutrients

In the attempt to dynamically model biological processes for carbon removal Henze *et al.* (1987) presented the Activated Sludge Model No.1 (ASM1). Subsequently there have been a number of developments to include also phosphorous removal. The group of such models is commonly referred to as ASM family. A profound background on the development and applications of these models is given in Henze *et al.* (2008).

A specific characteristic of the ASM approach lies in the representation of its state variables which are referred to as fractions. The COD fractions used within the ASM1 are shown in figure 2.11. It is furthermore important that that the model approach simulates carbon as COD equivalents. Therefore the conservation of mass as basis for the related equation system is based on COD mass balance Henze *et al.* (2008, p. 61).

The allocation of fractions within a model application of ASMs are wastewater specific.

The mathematical framework of the ASM approach is shown in a generalized manner in table 2.8. The conceptual framework behind lies in the application

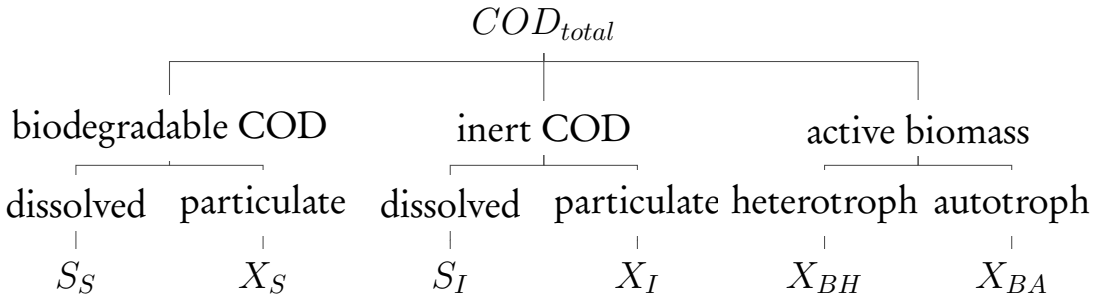


Figure 2.11: COD Fractions of ASM1

Table 2.8: Representation of stoichiometric matrix for an extensive reaction system (Gujer, 2008, p. 73)

j	Process	Materials i C1 $[M_1]$	C2 $[M_{i*}]$	C3 $[M_i]$	Process rate ρ_j $[M_{i*}L^{-3}T^{-1}]$
1	Process name 1	$\nu_{1,1}$	$\nu_{j,i}[M_i M_{i*}]$		$\rho_1 =$ Kinetic expression 1
2	Process name 2	$\nu_{2,1}$			$\rho_2 =$ Kinetic expression 2
3	Process name j	ν_j	$\nu_{j,i*}$	$\nu_{j,i}$	$\rho_j =$ Kinetic expression j
	Transformation rate r_i	r_1	r_{i*}	r_i	$r_i = \sum_j \nu_{j,i} \cdot \rho_j \quad [M_i L^{-3} T^{-1}]$

Table 2.9: Kinetics and stoichiometrics for carbon removal (Olsson and Newell, 1999, p. 60)

Process	Components Substrate	Oxygen	Biomass	Kinetics
Aerobic heterotrophic growth	$-\frac{1}{Y_H}$	$\frac{Y_H - 1}{Y_H}$	1	$\hat{\mu_H}\left(\frac{S_S}{K_S+S_S}\right)\left(\frac{S_O}{K_{OH}+S_O}\right)X_H$
Decay of heterotrophs	$1 - f_P$		-1	$b_H X_H$

of the so called Gujer Matrix[4] (Henze *et al.*, 2000) wherein the interrelation between an arbitrary number of processes can be described. Each process is defined according to its products and educts (i.e. C) in terms of its stoichiometric portion (ν) and a process rate (ρ).
An exemplary application of the Peterson Matrix is given in table 2.9 for carbon removal expressed by the two processes of aerobic heterotrophic growth and decay of biomass (Olsson and Newell, 1999, p. 59-60). Included materials are substrate (S_S), oxygen (S_O), and heterotrophic microorganisms (X_H).
In table 2.9 the constant Y_H represents the yield of heterotrophic biomass which represents the ratio of build up biomass related to consumed sub-

[4]The respective matrix notation used to be referred to as Petersen Matrix according to Petersen (1965).

strate. Furthermore $\hat{\mu_H}$ and K are the maximum growth rate of heterotrophic biomass and the half saturation constant of the Monod Kinetic equation. Furthermore f_P and b_H are parameters related to the decay of heterotrophic biomass. Some parametsrss above are wastewater specific and must be derived from wastewater analysis (e.g. growth rate).
Whereas table 2.9 shows the stoichiometry and reactions kinetics of the two processes at a single time step equation 2.26 and 2.27 show the evolution of the concentration of biomass and substrate over time. Each of the two equations describes the concentration as function of inflow minus outflow (q_{in}, q_{out}) as well the rate of change (r) within the reactor volume (V). This model behavior achieved thereby is characterized as a 0-d model since effects of defined processes is only described for a CSTR (cf. equation 2.2).
Eventually the rate of change for the state variable substrate (S_S) is shown in equation 2.28. Thereby the stoichiometric constants for the two processes relate to 1 and -1 respectively.

$$\frac{d}{dt}(VX_H) = q_{in}X_{H,in} - q_{out}X_H + r_H V \tag{2.26}$$

$$\frac{d}{dt}(VX_S) = q_{in}S_{S,in} - q_{out}S_S + r_S V \tag{2.27}$$

$$r_H = \hat{\mu_H}\left(\frac{S_S}{K_S + S_S}\right)\left(\frac{S_O}{K_{OH} + S_O}\right)X_H - b_H X_H \tag{2.28}$$

A list of further processes expressed in the Gujer Matrix such as nitrification and denitrification are given in section A.2.

2.4 Knowledge Representation and Reasoning

2.4.1 Introduction

Whenever the objective is to reflect a real world phenomena by some abstract model whether on a piece of paper or on a computer Knowledge Representation (KR) is inevitably to be applied. In this section terms related to knowledge representation and reasoning being the fundament of this work are defined. It starts with the question what is knowledge representation and reasoning all about and what can it be used for. It will be shown that knowledge representation in the field of computer sciences is in most cases based on concepts of logic and result in some kind of ontology. Since there are many ways of representing knowledge the main directions are outlined at the end of this section. The explanations within this section have been compiled according to the follow-

ing sources: Brachman and Levesque (2004); Davis *et al.* (1993); Hölldobler (2003); Russel and Norvig (2003); Sowa (2000).

What is meant by the term knowledge representation in the field of computer science can be explained by looking at the two involved nouns individually. A piece of *knowledge* can be described as the relation between an entity and a proposition. For simplicity propositions are abstract entities which can only be true or false. The term knowledge basically refers to the reasoning and interpretation of information (Hamouda *et al.*, 2009). A *representation* is a relation between two domains where the first stands for the second. *Knowledge representation* is then the field of study concerned with using formal symbols to represent a collection of proposition believed by some agent. *Reasoning* is the formal manipulation of the symbols representing a collection of believed propositions to produce representations of new ones. Logical conclusions of this kind are called logical inference (Brachman and Levesque, 2004, p. 2-4).

The objective in applying methods of knowledge representation and reasoning is to construct systems that behave according to the symbolic representations which have been implemented. Such systems are referred to as Knowledge-Based System (KBS) and the symbolic representations involved are referred to as Knowledge Bases (KBs). So what makes a system knowledge based, is not the type of logical formalism used or the fact that what it believes is true or not but the presence of a collection of symbolic structures representing what it believes and reasons with during the operation of the system. A further important property of symbolic representations within a knowledge base is that they can be understood from outside as standing for propositions. A necessary requirement for an effective knowledge based system is a strict separation between the knowledge base and the knowledge processing.

The knowledge based approach offers a number of desirable features. Since the behavior of such a system is determined by what it believes the behavior can be extended by adding new clauses to the knowledge base. In contrast, new tasks can be processed using a previously encoded knowledge base. Beyond functional advantages a knowledge base offers a platform for domain experts due to an unambiguous representation of the entities and relations between them within a represented domain. The possibility to reason upon the represented knowledge enables furthermore checks for consistency to locate erroneous beliefs. But the major advantage through reasoning is gained by the possibility that a system behaves according to the believes and not solely by what is ex-

plicitly represented. This means that through reasoning implicit knowledge can be revealed.
The latter introduced definitions have been summarized by Davis *et al.* (1993) as the five principles of knowledge representation (Sowa, 2000, p. 134):

1. **A knowledge representation is a surrogate.** Physical or abstract objects, events and relationships can not be handled directly on computers and are therefore represented by symbols that serve as surrogates. These symbols and relation form a model for external systems. Then a computer program can simulate the external system or reason about it by manipulating the internal surrogates.

2. **A knowledge representation is a set of ontological commitments.** For a database or knowledge base, ontology defines the categories of things that exist within an application domain. Oppositely, whatever is not defined in the ontology cannot be reasoned about. Categories within the ontology represent the so called *ontological commitments* of the designer, or so called knowledge engineer. Such a commitment to a common ontology is a guarantee of consistency but not of completeness (Ceccaroni, 2001, p. 24).

3. **A knowledge representation is a fragmentary theory of intelligent reasoning.** To enable reasoning it is necessary to define not just categories but also behavior and interactions within a domain. Such a description constitutes a theory of the application domain. This theory can be stated as explicit axioms or may be compiled directly into an executable program.

4. **A knowledge representation is a medium for efficient computation.** Proper encoding of knowledge is required to technically facilitate reasoning. This implies that knowledge is formalized in a language understandable by a computer.

5. **A knowledge representation is a medium of human expression.** A good knowledge representation language should facilitate communication between domain experts. Unlike conventional computer programs where knowledge is encoded within the program structure the knowledge in a knowledge-based system is separated from the processing part.

From the above listed principles it becomes clear that knowledge representation can be understood as a concept of programming regardless of a specific programming language. To further distinguish the concepts of knowledge-based programming from classical programming the following differences can be stated (Baader, 1996). Conventional programming has the objective to construct systems that are tailored to a specific application problem. The knowledge about the application domain is implicitly implemented in the structure of the program and hence can not be accessed or changed during application. In contrast knowledge-based programming has the objective to construct systems where the knowledge is represented explicitly using an appropriate representation formalism requiring no further information on how the knowledge is eventually processed.
In the context of KBS there are a number of other commonly used terms such as Expert System (ES), rule-based expert system, and Decision Support System (DSS). To delimit or to equate the meaning of those terms related to KBS some definitions are given below. Expert systems are knowledge based systems which emulate human reasoning using knowledge within a particular discipline (Heller *et al.*, 1998; Neumann, 2003). If the applied knowledge base is build by a set of rules ("if-then") the resulting ES is termed rule-based expert system. A decision support system is an information system that supports a user in choosing a consistent, near optimum solution for a particular problem in a reduced time frame (Hamouda *et al.*, 2009; Poch *et al.*, 2004).
Up to now attention has been paid to the question what knowledge representation is. In the following the focus is directed to the questions how knowledge can be represented and how a resulting knowledge base can be used. Knowledge representation in the field of computer science is intrinsically tied to the concepts of logic. Conventions found within computational logic offer what is needed to actually represent knowledge in a way to use this knowledge for logical entailment. The following section gives an insight on how concepts of logic are used for knowledge representation and reasoning.

2.4.2 Representing knowledge in Logic

Logic is the study of entailment relation-languages, truth conditions, and rules of inference and is therefore the key to represent knowledge (Brachman and Levesque, 2004, p. 11 et sqq.). In other words a particular logic offers a formal language for representing knowledge (by defining syntax and semantics) and a proof theory to enable reasoning about the represented knowledge. A formal logic is furthermore required to enable the computerized use of the

representations. According to Hölldobler (2003, p. 1) computational logic has the objective to model logical entailment providing theoretical foundations up to a technical realization. Therefore the ideas of *formalization*, *calculus formation*, and *mechanization* are combined. The idea of formalization constitutes that it is possible to solely draw conclusion upon concerned clauses regardless of their meaning. Logic of this kind is therefore also referred to as *formal logic* (syn.: symbolic logic). This agrees with the first principle of KR where a surrogate or more specific a symbol stands for something else. Calculus formation describes a system of rules which define how those symbols can be related to unfold new ones. Finally mechanization enables the propagation of logic symbols and rules to a computer. Therefore standardized machine understandable languages have to be used and inference engines must be provided which are able to solve the concerned calculus.

As true for natural languages there are different forms of logic (i.e. representation languages) varying in their degree of expressiveness. Regarding representation languages expressiveness itself is dependent on the ontological relation as well as on the epistemological relation. The *ontological relation* determines what is implied by a language in view of described reality. Propositional logic envisions reality being solely compound of facts which can be true or false. Thereby no propositions about sets can be made and therefore propositions logic allows no representation of any generalization. Predicate logic, also referred to as first-order logic goes beyond the assumptions of propositional logic and additionally realizes objects and relations between them. Temporal logic being a very distinct form of logic is able to distinguish facts at different times. The *epistemological relation* describes the states of knowledge which can be differentiated by a representation language. Common forms of logic such as propositional and predicate logic can only differentiate between the states false, true or unknown. Within fuzzy logic the truth-value of a fact can range between 0 and 1 depending of a considered model. Oppositely using a probability-theory any fact has an assigned degree of belief ranging from 0, improbable to 1, certainty. An overview of five forms of logic regarding the ontological and epistemologically relation is given in table 2.10.

As stated in the introduction a knowledge base is constructed by symbolic representations (i.e. propositions). Thereby *syntax* defines the used alphabet as well as the rules to constructs words and sentences (similar to grammar in natural languages). In other words syntax defines the inner structure of a language. *Semantics* deals with the meaning of what is expressed by syntax.

Table 2.10: Formal representation languages (Russel and Norvig, 2003, p. 308)

Language	**Ontological relation**	**Epistemologically relation**
Propositional logic	facts	true/false/unknown
Predicate logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, time	true/false/unknown
Probability-theory	facts	degree of belief $\in$ [0,1]
Fuzzy logic	facts with a degree of truth $\in$ [0,1]	known interval

Therefore semantics of a language provides the interpretation of what is represented. For a logical representation languages semantic defines the truth-value of a single proposition in a given context. For example, the algebraic proposition $x + y = 4$ is true for a given context where x equals 2 and y equals 2, oppositely the proposition is false in a context where x and y are equal to 1. A particular context is also referred to as a *model*. Thereby a model is simply a mathematical abstraction defining if a particular proposition is true or false. Semantics itself can be differentiated into various types of semantics. *Intended semantics* reflects what has been meant by the creator of a proposition or a complete knowledge base (e.g. expressed in natural language). Through a *formal semantics* a proposition can be formally described (e.g. by algebraic notation). Procedural semantics formalizes the way a representation is to be processed.

After the structure as well the rules for interpretation are defined the principles of logical reasoning can be introduced. The basis for reasoning is *logical consequence* (syn.: logical entailment) between two propositions. Logical consequence is represented by the notation $\vdash$. The expression $\alpha \vdash \beta$ states that the proposition β is a logical consequence from proposition α. The exact translation is: $\alpha \vdash \beta$ is true, if for any model where α is true also β is true. Logical consequence is applied in the process to draw logical conclusions from a knowledge base (i.e. a collection of propositions), this processing is called *logical inference*. A formal inference with a correct inference procedure guaranties that only correct propositions are deducted from other propositions. A second property of an inference algorithm besides its correctness is the completeness. An inference algorithm is complete if any proposition can be deducted from a given knowledge base.

Following this general introduction to logic the focus is now directed to the representation logic referred to as Description Logics (DLs) which has been applied within this work to codify knowledge in the domain of wastewater

treatment devices. Description logics is a subtype of first order logic (FOL) (see appendix B) which in turn builds up on propositional logic.

2.4.2.1 Description Logics

In the attempt to find a formal language which can be used for knowledge representation it is necessary to use a logic based language because only logic based languages allow reasoning upon represented knowledge.

In particular the choice for an appropriate logic based language is decided in finding the balance between a high expressiveness of such a language on one side and the decidability of its formal representations (Russel and Norvig, 2003, p. 437).

FOL based languages are semi-decidable, this means that every formula or its negation can be proofed true/false given a finite time. In practice this is still not applicable and hence FOL based languages are referred to as undecidable within practical applications. In contrast propositional logic offers only a low level of expressiveness not satisfying the demand of profound KR.

DLs are a compromise between propositional logic and FOL. A description logic language is always a trade-off between allowing expressiveness and maintaining decidability. Thereby all description logics systems belong to a family called concepts languages since the basis of these languages lies in the definition of concepts that can be generalized and specialized.

"The name description logics is motivated by the fact that [...] the important notions of the domain are described by concept descriptions, i.e., expressions that are build from atomic concepts (unary predicates) using the concept and role constructors provided by the particular DL" (Baader *et al.*, 2003)

The development of DLs has been influenced from structured inheritance networks (i.e. semantic networks) and KL-ONE. This development tried to overcome ambiguities in semantic networks and frames that were due to their lack of a formal semantics.

Most implementations of inference engines deal with subset of first order logic: e.g. description logics and horn logic (Raffeiner, 2005, p. 88 ff.)

A particularity of DL based knowledge bases are that they consists of a so called TBOX and an ABOX Sowa (2000, p. 7). Thereby the Terminological Box (TBOX) contains terminological knowledge and provides the base vocabulary of a domain. It defines hierarchies of concepts and relations between them. The Assertional Box (ABOX) contains assertional knowledge. This state properties of instances using the base vocabulary.

Table 2.11: Valid operators in DLs (Angeli, 2007; Brachman and Levesque, 2004)

$C \sqcap D$	union of concepts
$C \sqcup D$	intersection of concepts
$\neg C$	negation of concepts
$\forall R.C$	qualified universal role restriction
$\exists R.C$	qualified existential role restriction
$R \sqsubseteq S$	inclusion of roles (subsumption)
$R \sqcap S$	union of roles
$R \sqcup S$	intersection of roles
$\neg R$	negation of roles
R^{-1}	complement of roles

Syntax and Semantics

Logical symbols in DLs are similar to those in FOL. First of all there are *concepts forming operators* such as $\forall$, $\exists$, $\sqcup$, and $\sqcap$. Second, there are connectives $\sqsubseteq$, $=$, and $\rightarrow$. Furthermore *punctuations* and *positive integers* can be used. An overview on possible operators of DLs is given in table 2.11.
Besides logical symbols there are non-logical symbols (i.e. building blocks) used in DLs (Brachman and Levesque, 2004, p. 158) .
Therefore DL is also known as *terminological logics* (Ceccaroni, 2001, p.36) since DL describes knowledge in terms of concepts and role restrictions.
Concepts can be understood as unary predicates. Thereby a concept is a set of objects. In turn roles can be understood as binary predicates since they unite two objects (Angeli, 2007, p. 19). Concepts can be further distinguished into atomic and complex concepts. Thereby complex concepts are formed by two or more atomic concepts. An overview of the building blocks is given below (Tessaris, 2001, p. 18) (example taken from Angeli (2007)):

concept (syn: class) describe unary predicates

> **atomic concept** Example: `University`, `Person`
>
> **complex concept** Example: `Student` $\equiv$ `Person` $\sqcap \exists$ `studiesAt University`

object (syn.: individual, instance, constant); Example: `tom`, `tu dresden`

role (syn.: property, relation) describe binary predicates; Example: the role `studiesAt` can be used in the formal sentence
`studiesAt(tom, tu dresden)`

Table 2.12: Syntax and Semantics (Tessaris, 2001)

	Syntax	Semantics	Description
concept expression constructors	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	primitive concept (atomic concept)
	$\top$	Δ^{I}	main concept
	$\bot$	$\varnothing$	empty set
	$\neg A$	$\Delta^{\mathcal{I}}\ A^{\mathcal{I}}$	general negation (NOT)
	$A \sqcap B$	$A^{\mathcal{I}} \cap B^{\mathcal{I}}$	conjunction (AND)
	$A \sqcup B$	$A^{\mathcal{I}} \cup B^{\mathcal{I}}$	disjunction (OR)
role expression constructors	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	role name
	$R \sqcap P$	$R^{\mathcal{I}} \cap P^{\mathcal{I}}$	role conjunction
	$R \sqcup P$	$R^{\mathcal{I}} \cup P^{\mathcal{I}}$	role disjunction
	R^{-1}	$R^{\mathcal{I}} \cup P^{\mathcal{I}}$	role complementation

Reasoning, Inference and queries

"Queries on a DL KB are answered by an inferential process involving both Tbox and Abox parts. The answer to a query is deduced as logical consequences of the content of the KB according to the formal semantics" Tessaris (2001). DL based systems offer the following types of logical consequence (Tessaris 2001, Brachman and Levesque 2004, p. 179-181): consistency checking, satisfiability checking, subsumption, instantiation.
Thereby consistency checking checks if a given KB in DL includes no contradicting assertion (including TBox and ABox).
Second, the analysis of satisfiability of concepts checks if for any concept direct instances can exist. If not the overall KB is inconsistent. In particular a concept C is satisfiable for a model $\mathcal{I}$ within the context of a given TBOX ($\mathcal{T}$) if $C^{\mathcal{I}} \neq \bot$.
Third, subsumation (syn.: classification) derives class hierarchy between concepts. A concept C is subsumed by concept D if $C^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ is valid for any model $\mathcal{T}$.
Forth, instantiation (syn.: realization) derives the direct concept of each instance.

Families of Description Logics

There are various types of DLs varying in their expressiveness. The expressiveness varies according to the number of contractors provided by the particular DL (Tessaris, 2001). In order to simplify the notation of different types of DLs Schmidt-Schauss and Smolka (1991) proposed a system of symbols to organized different types of DLs. A short overview on types of DLs is given in table 2.13.

Table 2.13: Types of DLs (Schmidt-Schauss and Smolka, 1991)

Operator/Constructor	Syntax	Language	
conjunction	$A \sqcap B$	$\mathcal{FL}$	$\mathcal{S}*$
value restriction	$\forall R.C$		
existential quantifier	$\exists R$		
top	$\top$	$\mathcal{AL}*$	
bottom	$\bot$		
negation ($\mathcal{C}$)	$\neg A$		
disjunction	$A \sqcup B$		
existential restriction	$\exists R.C$		
number restriction ($\mathcal{N}$)	$(\leq nR)(\geq nR)$		
set of individuals ($\mathcal{O}$)	$a_1, ..., a_n$		
hierarchy of relations	$R \subseteq S$	$\mathcal{H}$	
inverse	R^{-1}	$\mathcal{I}$	
qualified number restriction	$(\leq nR.C)(\geq nR.C)$	$\mathcal{Q}$	

As an example of a particular DL $\mathcal{SHOIN}^{(\mathcal{D})}$ as W3C standard OWL(DL) includes the following language constructs.
Concepts can be defined through nominal values, indicated through the symbol $\mathcal{O}$. As example the concept `Color` may be defined through three available colors as $\equiv$ {`red`, `blue`, `green`};
This concept definition is also referred to as *closed classes* or *one of*. In particular a class is defined by a restricted list of individuals. This can be expressed in DL as $C = \{a, b, c\}$ and in FOL as $(\forall x)(C(x)) \leftrightarrow (x = a \wedge x = b \wedge x = c)$.
$\mathcal{SHOIN}^{(\mathcal{D})}$ furthermore allows for role construction through sub properties and inversion of roles. Sub property of roles are expressed in DL as $R = S$ and in FOL as $(\forall x)(\forall y)(R(x, y) \rightarrow S(x, y))$. As example the role `hasSon` is a sub property of `hasChild`.
Inversion of roles ($\mathcal{I}$) can be for example the role `hasChild` and its inverse role `hasParent`. This is expressed in DL as $R = S^{-1}$ and in FOL as $(\forall x)(\forall y)(R(x, y) \leftrightarrow S(y, x))$. Role transitivity is expressed as $R \sqsubseteq +R$ and in FOL as $(\forall x)(\forall y)(\forall z)(R(x, y)) \wedge R(y, x) \rightarrow R(x, z))$.
Furthermore $\mathcal{SHOIN}^{(\mathcal{D})}$ allows for the definition of equal concepts by the specification of *same as*. That defines that two individual names refer to the same element ($a = b$).

2.4.3 Representing knowledge in Ontologies

2.4.3.1 Introduction

The meaning of the term ontology as well as the use of the concept behind is slightly diffuse. Even so the concept of ontologies is often the origin behind the structuring of knowledge in everyday life. Simplified there are two

Table 2.14: the expressivity is encoded by a letter (Schmidt-Schauss and Smolka, 1991)

Letter	Expressiveness
$\mathcal{F}$	Functional properties
$\mathcal{E}$	Full existential qualification (Existential restrictions that have fillers other than `owl:thing`)
$\mathcal{U}$	Concept union.
$\mathcal{C}$	Complex concept negation.
$\mathcal{S}$	An abbreviation for $\mathcal{ALC}$ with transitive roles.
$\mathcal{H}$	role hierarchy (sub properties - `rdfs:subPropertyOf`).
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.
$\mathcal{O}$	Nominals. (Enumerated classes of object value restrictions - `owl:oneOf`, `owl:hasValue`).
$\mathcal{I}$	Inverse properties.
$\mathcal{N}$	Cardinality restrictions (`owl:Cardinality`, `owl:MaxCardinality`).
$\mathcal{Q}$	Qualified cardinality restrictions
$(\mathcal{D})$	Use of datatype properties, data values or data types.
$\mathcal{AL}$	Attributive language. This is the base language which allows: • Atomic negation (negation of concepts that do not appear on the left hand side of axioms) • Concept intersection • Universal restriction • Limited existential quantification

meanings of ontology. First, in philosophy it regards the study of being as a branch of metaphysics. In other words, ontology in philosophy is the study of existence, of all the kinds of abstract and concrete entities that make up the world. And second, being the focus in this work, it refers to the theory concerning the kinds of entities that are admitted to a language system within the field of AI in particular in the field of knowledge sharing and reuse (Raffeiner 2005, p. 3-5, de Bruijn 2003, p. 4). Thereby ontologies are a way to represent and store knowledge about a certain domain in a declarative manner (i.e. facts are directly represented by symbols and not encoded as procedural programs) (Raffeiner, 2005, p.13). Ontologies can be understood as an organizing principle (Raffeiner, 2005, p. 88), or a model of a particular domain.

In the context of knowledge representation a common used definition of the term ontology is cited by Gruber (1993b): "An ontology is a formal explicit specification of a shared conceptualization." Thereby conceptualization refers

to an abstract simplified view of the world represented for some purpose. Through conceptualization (syn.: abstraction) relevant concepts of a phenomena are identified. Furthermore only a certain domain of interest (or universe of discourse) is covered. A conceptualization provides a coherent vocabulary for representing and communicating knowledge about a domain, but not the representational structure Bogusch *et al.* (2001, p. 965). The term specification in the above definition is about the chosen concrete form. Related to the specification the term formal denotes that the chosen form must be machine understandable (Ceccaroni, 2001). The term explicit in the above given definition relates the fact that concepts and relations are explicitly defined. Explicit means that all terms regarding the domain are defined within the ontology (no additional knowledge is required). Descriptions at the knowledge level can be divided into a conceptualization and a formalization. While a conceptualization consists of the entities that are assumed to exist in the world and their interrelationships, the formalization of knowledge entails the representation of knowledge about the domain as sentences in a formal language (Bogusch *et al.*, 2001, p. 965). Finally shared means that ontologies commonly capture consensual knowledge which is accepted by a relevant part of the scientific community.

It can be concluded that a particular ontology can be understood as a result of knowledge representation. The process of defining an ontology is highly intuitive (analogy to model development). The process is referred to as ontological engineering. Furthermore ontologies can be encoded in different forms and languages. In this work logic will be used. Another possibility is the use of graphs in particular semantic networks (Russel and Norvig, 2003, p. 433).

In the context of ontology there are various other commonly used terms such as meta data, database and relational database. To delimit or equate the meaning of these terms related to the meaning of ontology some definitions are given next. Meta data (data about data) can be any kind of data that gives additional information about another set of data (e.g. regarding the structures of a relational database) or administrative information (e.g. access rights). Meta data not necessarily means to go to an upper level but outside more like another semantic layer (Raffeiner 2005, p. 4, Brase 2005). Regarding the concept of a database it must be stated that ontologies do have a function similar to database schema (Ceccaroni, 2001, p. 15). However the syntax and semantics of ontologies is richer. Ontologies use a consensual terminology in contrast to databases. Furthermore ontology provides a domain theory in contrast a

database provides the structure of a data container. Relational database management systems (RDMS) (Raffeiner, 2005, p. 30) have a fixed schema which cause problems to represent complex structures. Ontologies don't have a fixed schema (ontologies also use a fixed schema but this is not subject to its constraints). In fact fixed schema provide an efficient organization but come with the cost of in-flexibility (Sowa, 2000, p. 488).

2.4.3.2 Classifying ontologies

In the attempt to grasp the concept of ontology it is useful to explore by which properties ontologies can be differentiated and systematized. The main characteristics of an ontology are generality, expressiveness and formality (de Bruijn, 2003). The characteristics may better be expressed by pairs of opposition such as generality vs. specialization, lightweight vs. heavy weight, and informal vs. formal. The first refers to what is actually described by a particular ontology. More precise, the content ranges between the two extremes of generality and specialization.

The second measures what can actually be expressed through a particular ontology. Lightweight ontologies include concepts, properties that describe concepts relationships between concepts and concept taxonomies. Heavyweight ontologies also include axioms and constraints and hence enabling reasoning (de Bruijn, 2003, p. 7).

The third refers to the strictness of how a particular ontology is described ranging from formal to informal representations (Gruber 1993a, Ceccaroni 2001, p. 27). Informal specification are basically structured information using a natural languages. Semi-formal ontologies describe knowledge in an abstract formal language. And through formal specification the definition of terms is achieved with formal semantics and proof theory.

Commonly generality is differentiated according to three levels (Guarino, 1998): top-level ontologies, domain ontologies, and application ontologies. These three levels may also be termed upper, middle and lower ontologies (Raffeiner, 2005, p. 23-24). Top-level ontologies are also referred to as generic or general ontologies. They have the highest level of generality. They are domain-independent and specify general knowledge or common sense prototypical knowledge regarding (in)tangible subjects, space, time (concepts that are basic for human understanding).

The requirement for a top-level ontology is that they are applicable for any low level ontology. This implies that facts are generalized to allow reasoning across ontologies (Russel and Norvig, 2003, p.399). Using top-level ontology

follows a top down approach where general concepts are used to describe particular concepts. Example are the OpenCyc Ontology and the Suggested Upper Merged Ontology (SUMO). The use of top level ontologies was intended by philosophy.

Domain ontologies or task ontologies describe generic concepts of a particular domain (e.g concepts on material, weather conditions, plants, animals). A remarkable example out of the domain of chemical process modeling is the ontology OntoCAPE (Marquardt *et al.*, 2010).

The lowest level of generality is formalized in so called application ontologies. They describe concepts depending on a particular task and domain (e.g. for the use of a particular software application).

Out of the context of generality there are also so called representational ontologies. Representational ontologies describe representational entities without describing what they describe (knowledge on knowledge or meta data).

Regarding the expressiveness of ontologies a list of examples is given in table 2.15 starting with types of ontologies with the lowest expressiveness (de Bruijn 2003, p. 7; McGuiness 2003).

Table 2.15: Various types of ontologies with different degrees of expressiveness

Controlled vocabulary	expressiveness ↓	a list of terms; used to classify resources; used to prevent authors choose other terms (Raffeiner, 2005, p. 5)
Thesaurus		relations between terms; is an extension of taxonomies; ISO standard ISO 2788 (regarding monolingual thesauri) and ISO 5964 for (multilingual thesauri) (Raffeiner, 2005, p. 5)
Informal taxonomy		there is an explicit hierarchy; extension of a controlled vocabulary; arranges terms in an hierarchy
Formal taxonomy		uses strict inheritance
Frames		a frame contains a number of properties and these properties are inherited by subclasses and instances
Value restriction		values of properties are restricted
General logic constraints		values may be constraints by logical or mathematical formulas using values from other properties
First-order logic constraints		very expressive ontology language; allows for reasoning

2.4.3.3 Ontology construction

The development of an ontology is a very intuitive process which requires profound understanding on the domain to be modeled as well a deep comprehension of the language used for knowledge representation. Thereby the development process is referred to as ontology engineering. This section presents general information on ontology engineering.

Ontology Engineering

Because the conceptual framework of formal ontologies for knowledge representation provide only the means for representation there are no precise methods on the actual steps required to build an ontology (Russel and Norvig, 2003, p. 328).

However the development process of a domain or application ontology is very similar to the development process of a process model (formalized in mathematical notation) (see section 2.5.5). It all starts with the definition of an objective (What will the ontology be used for?), followed by the step of conceptualization, abstracting those concepts from reals world which are relevant in the desired model objective. In a next step the conceptual model must be translated into a formal language for knowledge representation (e.g. OWL(DL)). Eventually the model (i.e. ontology) must be validated before it can be applied (Blomqvist 2009; Fernándes-López and Gómez-Pérez 2000; Antoniou and van Harmelen 2004, p. 205).

In short the overall process of building a (formal) ontology covers the two fields of (i) conceptualization and (ii) formalization. The first step is primarily conducted by domain experts whereas the second step is primary conducted by so called knowledge engineers.

The overall development process should be aligned to a number of design principles of ontologies.

Analogue to the five principles of knowledge in general as described in section 2.4.1, Gruber *et al.* (1995) defined a list on design principles related to ontology engineering. These principles state that the development of an ontology must aim on: clarity, coherence, extendibility, minimal encoding bias, and minimal ontology commitment.

Thereby clarity states that the definition of terms should be objective (i.e. be independent from context) and whenever possible the definition should be achieved by logical axioms. The term coherence simply states that terms defined (and those that can be inferred) within a KB must be consistent. Ex-

tendibility indicates that the design of an ontology should always allow for the later addition of further concepts while keeping consistency. The second last requirement aims at a codification with the least commitment to a particular serialization format or application to be able to use the ontology in different applications. The last requirement, minimal ontology commitment, states that concepts within an ontology should be defined with the least claims as possible about the world in general. The idea behind this lies in the fact that less ontological commitment increased the extendibility of an ontology. In combination with the first claim the aim can be summarized to describe a concept as precise as possible by using as less as possible axioms.

Although there is no single method for optimal ontology development there are a list of various methods proposed.

Detailed step by step methods for ontology development have been presented in the following publications: Gruninger *et al.* (1995); Mizoguchi (2004a,b); Noy and McGuinness (2005); Uschold and King (1995). More general comment on ontology engineering are given by Russel and Norvig (2003); Sowa (2000).

Closely related to the process of ontology engineering is the term ontology design pattern (syn.: ontology pattern). According to Blomqvist (2009): "An ontology pattern is a set of ontological elements, structures or construction principles that intend to solve a specific engineering problem and that recurs, either exactly replicated or in an adapted form, within some set of ontologies, or is envisioned to recur within some future set of ontologies." Further details on ontology patterns are described in Gleich (2008); Hoekstra (2009).

Categories and Objects

As stated in the section on DL (section 2.4.2.1) the main building blocks of a formal ontology represented in DL are concepts (syn.: class or category) and objects (syn.: individual or instance). In this context ontology representations are called identity-based technologies because they focus on things, called concepts or subjects of discourse (Raffeiner, 2005, p. 15).

It follows that a fundamental question in developing formal ontologies is whether to represent an entity as concept or as object. The decision is also referred to as reification (Brachman and Levesque, 2004, p. 41).

The decision to describe a concept by a class or an instance is a final decision because no other concepts can be describe below an instance (Raffeiner, 2005, p. 32).

For simple ontologies reification is a simple task. For instance for an ontology on staff members possible concepts can be `Employee` and a possible object could be particular persons such as `tom`. However for complex ontologies that may cover multiple domains the process of reification becomes much more complicated.

Turning to concepts it can be noted that reasoning is often achieved on the level of concepts (Russel and Norvig, 2003, p.400).

This is related to the fundamental idea of an ontology is the relation of concepts to more general concepts and oppositely to further specify concepts by refined concepts (generalization vs. specialization). The resulting concept hierarchies are also referred to as taxonomies.

Through categories the concept of inheritance is implemented thereby as a super-class inherits its specialization to its subclasses. Subclasses are related to each other by so called is-a relations.

Besides concept hierarchies concepts can further be defined as being disjunct from other concepts. This means that two concepts can be defined as being mutually different (e.g. the concept `Men` is disjunct from `Woman`, which means if an instance is member of `Men` it can not be member of concept `Woman`)

Especially for the representation of physical entities taxonomies can also be defined as is-part-of relations instead of is-a relations. The philosophical branch dealing with the description of physical compounds is also referred to as metrology (Sowa, 2000, p. 95). For instance the concept Car may be defined by its necessary parts such as `Engine` is-part-of `Car`. Sowa (2000, p. 107) differentiates three types of part of relations as being discrete, lumpy, and continuous.

The use of concepts for KR can be further specialized by looking on different role types of concepts. In this context Sowa (2000) lists three types of concepts: structural type, role type, and phenomenal type. For example does a concept Cube receives its attribute from its geometrical form. According to Sowa (2000, p. 80): "a structural type classifies an entity x by a monadic predicate that depends only on properties that can be observed in x itself". In contrast a role type describes an entity by a role in relationship to another entity (i.e. dyadic relation). A role type would be for example the concept `Student`. Thereby role types may change over time. The last concept type, phenomenal type depends on the internal form an an entity. The concept `HumanBeing` is a phenomenal type because it is independent of external relations and it will not change over time.

The role type can be further distinguished when looking on extrinsic and intrinsic properties (Sowa 2000, p. 87 and Russel and Norvig 2003, p. 406).
An *intrinsic property* is a property which things have in virtue of the way they themselves are as opposed to an *extrinsic property*, which things have in virtue of their relations or lack of relations to other things. Intrinsic properties can be inherited, extrinsic properties can not be inherited (Lewis 1986 and Sowa 2000, p. 87, 502). For example a piece of butter can be cut in half resulting in two pieces of butter. In contrast a pig can be parted but the result are not two pigs. Some examples of intrinsic properties are: density, melting point, taste of an entity. Some examples of extrinsic properties are: weight, length, shape of an entity.
Using this distinction Sowa (2000) defines the three sub types of the role type: composite, correlative, and component.
Composite entities have a relationship to each component within itself. Thereby the subtype a *whole* is made up of its parts and a *substrate* is the underlying material that supports the dependent properties such as size, weight etc.
A *correlative* holds a relationship to something outside itself and at the same time the counterpart holds a relation to the correlative entity (e.g. mother and child).
The last type, *component*, holds a relationship to the composite in which it inheres. Subtypes of of component are *part* and *property*. Thereby part is part of a whole but can exit without the whole (e.g. `Car` and `SteeringWeel`). In contrast a property can not exit without some substance.
A structured analysis of the latter introduced semantic analysis of concept hierarchies is also provided by the OntoClean method presented by Guarino and Welty (2002, 2009) in the paragraph below.

Ontology validation

Stating that an ontology is a model which describes some abstract or real part of the world (in a formalized, declarative manner) in analogy to formal process models the question arises of how to validate an ontology. Unlike to process models ontologies can not be calibrated and validated against measured data for example. However the correctness of an ontology may be checked within three levels. First of all the correctness of an ontology should be achieved through the development process conducted by a group of domain experts. That follows from the definition of an ontology according to Gruber (1993b) which states that an ontology captures consensual knowledge. That means that the ontology should be agreed upon a group of domain experts. Further-

Table 2.16: Overview on philosophical notions

Identity	How are instances of a class distinguished from each other?
Unity	How are all the parts of an instance isolated?
Essence	Can a property change over time?
Dependence	Can an entity exist without some others?
Permanence	How long do entities last?

more the logical correctness of a formal ontology can be checked by automatic reasoners that reveals possible logical conflicts from axiomatic description of concepts within a KB. Reasoners are part of some ontology editors such as Protégé (Horridge *et al.*, 2009). Furthermore the OntoClean method can be applied to check the semantic correctness of concept taxonomies. In particular OntoClean is a methodology for validating the ontological adequacy of taxonomic relationships (Guarino and Welty, 2002, 2009). It is based on highly general ontological notions drawn from philosophy, like essence, identity, and unity, which are used to characterize relevant aspects of the intended meaning of the properties, classes, and relations that make up an ontology (see table 2.16). For an application of OntoClean see also Herb (2006).

Ontology editors

As stated above the development process of an ontology can be roughly differentiated into a conceptualization phase and a formalization phase. Both phases can be supported by particular software tools. Tools applied for first step deal with semi-automatic and automatic ontology construction form text sources.

Tools that support the formalization step of ontology development are referred to as ontology editors. These kind of editors support the modeler in translating axiom definitions into formal ontological languages such as OWL. Furthermore do they provide for automatic consistency checkers through implemented reasoners. A prominent example of such an editor is Protégé[5] (Horridge *et al.*, 2009).

2.4.3.4 Representation formats & Ontology languages

An ontology is an abstract theory which can be represented by a number of representation formats (Raffeiner, 2005, p. 5). Thereby a representation format does not mean a specific serialization (type of notation) format of an ontology language.

[5] `http://protege.stanford.edu`

Examples for representation formats (developed solely in AI) are (Raffeiner, 2005, p. 6-7): KIF (Knowledge Interchange format, CGIF (Conceptual Graph Interchange Format), CYCL which is used in the CYC Ontology.

With the development of the World Wide Web there are new representation formats based on Standard Generalized Markup Language (SGML) and its extension Extensible Markup Language (XML). A very common one is the Resource Description Framework (RDF) with its ontological extension Web Ontology Language (OWL).

RDF and OWL have its roots in formal logic and mathematical graph theory. The historical influences on the development of OWL are depicted in figure 2.12.

These influences correspond to the layered architecture of logic based knowledge representation in formal ontologies. Thereby the important layers are in order of increasing significance: The first layer lies in the use of unicode and Uniform Resource Identifier (URI). This allows for a system independent way to encode and address information. The next layer is provided by XML. Thereby XML offers the standardized syntax to represent arbitrary information structures (serialization format). The next higher layer is RDF and Resource Description Framework Schema (RDFS). Thereby RDF allows for the representation of assertions as triples (<S,P,O> subject-predicate-object triple). Through tripes graphs of concepts and relations can be described as it is done in ontologies. The next upper level is defined by formal ontologies which allow to express advanced properties as needed for semantically qualify the contents of ontology.

The top level is defined though logic which allows for defining inference engines which can be used to derive assertions that are not expressed directly within a KB.

It is obvious that any formal language will restrict the scope and detailedness of knowledge that can be represented. However, there are many formal and semiformal languages which can be used to represent knowledge (e.g. UML, SQL) but they do not allow to draw logical consequences from what is represented as ontology languages.

The fundamental requirements on ontology languages such as OWL correspond to the development criteria of DLs. Thereby it is desired to achieve a high expressiveness and still guarantee decidability by automated inference (de Bruijn, 2003, p. 15).

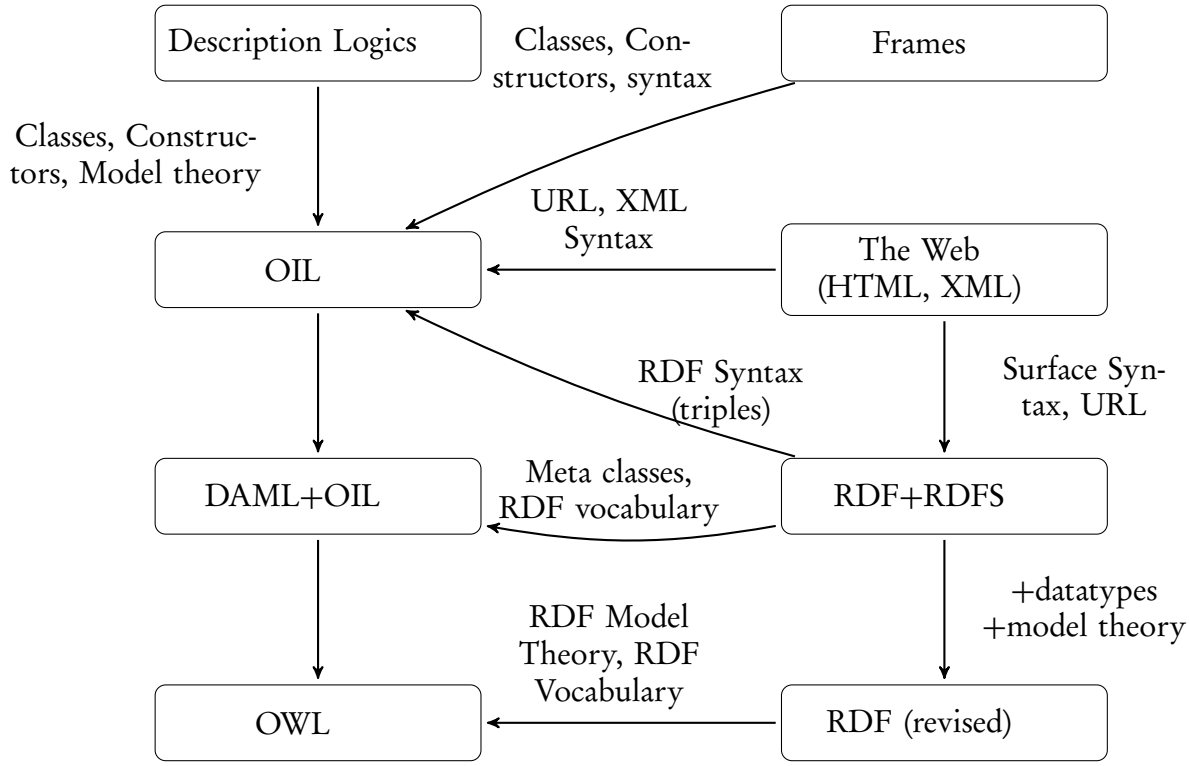


Figure 2.12: Influences on OWL (adopted from Patel-Schneider 2004)

2.4.4 Possibilities, limitations, and trade-offs

The review within this section gave an overview on the principles of knowledge representation in particular through the used of logic based languages. This section now will summarize the main advantages and disadvantages of DL based knowledge representations. The main advantages can be summarized as follows.

The main advantage lies in a separation between knowledge representation and knowledge processing. Thereby a DL based KB holds only declarative knowledge on the modeled domain by omitting notions an the use of the encoded knowledge. This improves the extendability of the KB as well as the reuse of the encoded knowledge. Thereby the possibilities for extending a KB are beyond those of conventional database due to the open data schema of ontology in opposite to conventional databases. Another example in using a DL based KB lies in the possibility to deduce hidden knowledge. Providing a well constructed DL based KB thereby allows for reasoning across specified axioms to deduce implicit facts. Furthermore, a DL based KB functions as an ideal intermediated between human understanding and computer based application of this knowledge. In short, formal ontologies can be used by humans as well as by computers. This is because concept based modeling is close to human

perception. Consequently the development of formal ontologies is more intuitive as for example the use of complex mathematical notations. A welcome side effect of a formal ontology is furthermore that it can serve as conceptual framework that supports the communication between domain experts (also across different domains).

In contrast there are a number of limitations accompanied by the use of formal ontologies. As true as for any other type of knowledge representation there is an inevitable loss of preciseness and even information of what is intended to be modeled and what is eventually by represented. This comes as described in the latter sections through the degree of expressiveness provided by the chosen representation language. Consequently the advantage of DL based languages (e.g. OWL(DL)) that is their decidability through reasoners comes to the cost of a limited expressiveness. As for FOL based languages in general the epistemological relation (i.e. states of knowledge) are only true, false, and unknown. It follows that DL based languages do not allow the capturing of degrees of truth (e.g. fuzzy logic) or degrees of beliefs or probabilities (e.g. probability theory).

The advantage of DL based languages to be able to represent objects and relations between them makes it difficult on the other side to represent arithmetic relations or even procedural knowledge.

Furthermore, languages for knowledge representation such as OWL(DL) have reached a high degree of standardizations through the use of widely used web specifications which promotes the reuse and extendability of developed ontologies. However the way knowledge is represented within an ontology always reflects the intention of the modeler. As result there may exist various different ontologies (i.e. declarative models) about a domain.

2.5 Process Modeling

2.5.1 Introduction

Process modeling is a common method in all engineering sciences. In this work the focus lies on process modeling in the domain of chemical process engineering in particular within the domain of wastewater treatment technologies. Process modeling in the domain of chemical engineering is included in the field of Computer-Aided Process Engineering (CAPE). Thereby CAPE is defined according to Braunschweig and Gani (2002, p. 3) as "The application of a systems modeling approach to the study of processes and their control, safety, environmental protection and utility systems as an integrated whole, for the viewpoints of development, design and operation."

Since most definitions and approaches regarding process modeling compiled in this section originate from contributions out of the field of chemical engineering an important difference regarding process modeling in the field of chemical engineering and wastewater treatment technologies must be noted. The objective of chemical engineering in general is the efficient production of desired materials and substances through known raw materials and reactions. And therefore process models can be developed including detailed information on substances and corresponding reaction kinetics. In contrast, the objective in applying wastewater treatment technologies is the purification of water by roughly known substances for possible water and raw material reuse. Due to the multitude of different substances within wastewater streams (especially for organic pollutants) no precise (i.e. substance/compound specific) reaction kinetics can be defined. Hence reaction kinetics for modeling treatment processes are usually wastewater specific. Aside this fact the methodologies of model development, model representation, and model applications are universally applicable.

Process modeling comprised the two main steps of *model development* followed by *model application*. The overall methodology is also referred to as *modeling process*. Within model development a model evolves in general through a number of model representations starting from conceptual informal representations up to a formal representation. Eventually a model is implemented into a modeling tool. Modeling tools offer some sort of solvers which enable the application of a model. It follows there from that a model can not be used by itself but can only be used in combination with a modeling tool. The expressiveness of a model is therefore dependent and restricted by the functionality offered by the used modeling tool.

2.5.2 Definitions & Terminology

This section defines relevant concepts related to process modeling and suggests a terminology which is used within this work. In the scope of this work process modeling is about methodologies (i.e. the development of a model for a particular objective) and different types of model representations. The main entities used are defined by the terms: model, modeling framework, and modeling environment. Any other terms are categorized below the latter introduced three main entries. The classification of terms in this section makes no claim on completeness or generality but is an attempt to promote the reader's comprehension.

Model

Similar to the definition of knowledge representation a model is a representation and is thereby standing for something else. According to Kuipers (1994, p. 321) a model is a description of some phenomenon, created for some purpose. There are many forms how a model can be represented. Issues related to model representation are entangled by the term model framework. The chosen representation is tailored by the modeling objective, the availability of an appropriate representation language and a corresponding modeling environment. In most cases a model follows thereby a Closed World Assumption (CWA). Thereby CWA indicates that all required functionality to be prescribed by the model objective is contained within the model itself. Regarding the definition of what a model is in process modeling some more aspects are discussed in the following publications: Bayer and Marquardt (2003); Bayer *et al.* (2000), Molitor (2000, p. 8), Marquardt (2005, p. 115), Hangos and Cameron (2001).

According to the objective of a model different types of application domain models can be distinguished Morbach *et al.* (2008b). One can differentiate between information models and ontologies. Commonly information modeling is applied for the analysis and formalization of information structures necessary for software design. Thereby a model evolves through the phases of a conceptual model over a design model resulting in an implementation model. Ontologies are generally applied to represent domain knowledge. Thereby ontologies and information models must not be regarded as exclusive.

Modeling Framework

The modeling framework provides the basic conceptual structure to represent a model for the use in a model environment (Wedel *et al.* 2002, p. 93-94 and Marquardt 1995b, p.602-605). In other words the model framework provides modeling concepts and an unambiguous formal language to denote the concepts (Steele, 1999). Thereby the modeling framework forms the link between model and model environment. Such a framework provides a set of concepts that can be used to construct process models. The framework must at least provide one form of representations of these concepts to enable the description of models. The overall framework is similar to an architecture that exploits the model representation and organizes modeling functionality in a tool implementation. As stated before the representation of a modeling framework

considered is with respect to two independent coordinates: a set of modeling concepts and an unambiguous formal language to denote the concepts.

Modeling Environment

The intention of a process modeling tool (syn.: modeling tool, modeling software) is to support a certain modeling methodology, where a methodology in general is commonly understood as a combination of a certain work process and a notation of domain-specific concepts (Wedel *et al.*, 2002, p. 106-107). The variety of possibilities of model applications within a single model environment is referred to as tool functionality. In most cases are modeling tools applied at the end of the modeling process: used for model validation and application where the application is in most cases simulation. Within the past decades numerous computer-aided tools have been developed to facilitate likely all possible aspects of process modeling. None of them is capable to fulfill all modeling needs, instead each of them is designed to address a number of particular needs. Existing computer-aided process modeling tools can roughly be characterized by the following approaches (Bieszczad, 2000, p. 25-34), see table 2.17.

More details on the classification on modeling approaches and tool architecture are given in the work of Wedel *et al.* (2002, p. 113) and Morbach *et al.* (2008a).

Finally it must be realized that a model is inextricably joined to the used model framework and furthermore that the model framework is depending on the used model environment. While discussing a particular model one must therefore clearly differentiate between the actual model objective, and its various representations.

Table 2.17: Types of modeling tools

Sequential modular flow-sheet simulators	Each process is depicted by a predefined unit operation from a library.
Programming languages	The entire modeling knowledge is directly embedded in the solution techniques of the corresponding numerical algorithms.
Spreadsheets	Similar to programming languages the model is directly implemented.
Equation-based process modeling tools	The model formulation is declarative and implicit by the use of mathematical conventions.
Phenomena-based process modeling tools	Process models are represented through the description of elementary physical and chemical phenomena.

2.5.3 Application of process models

There is a broad field of applications for process models in engineering sciences. The main areas are (Bieszczad 2000, p. 20, Olsson and Newell 1999, p. 31, Hangos and Cameron 2001, p. 7): design, optimization, analysis, control, scheduling, diagnosis and training. The latter mentioned application areas do intersect in many cases and hence serve only as a loose classification. In most applications process models are used for simulation of real phenomena meaning the propagation of state variables over time (Hocking *et al.*, 2002, p. 165-166).
In particular diagnosis enfolds the use of models to identify problems or misbehavior of production or treatment lines. Similarly scheduling and optimization is used to efficiently operate production lines or more general to achieve efficient system operation (Puigjaner *et al.*, 2002, p. 219). Within this work the focus is on process design (syn.: synthesis of process flow-sheets) (Eggersmann *et al.* 2002, p. 335-336 and Hocking *et al.* 2002). The field of process design is further discussed in section 2.6. For more detailed information on simulation, analysis and design see also Hocking *et al.* (2002).

2.5.4 Problems and Requirements

The use of process models within engineering applications has become an indispensable practice within the last decades. Looking back a number of problems and requirements related to the use of process models, model development and modeling tools can be identified (Braunschweig and Gani, 2002).
The development of a process model is a tedious process. It requires sufficient amount of sound data for validation and calibration. In most cases model development is done by experts whereas models are applied by a much broader group of end-users which may not always be familiar with the behold model complexity. Furthermore it is usually not possible to integrate all existing expert knowledge from a particular domain from engineers and scientist into a particular model. On the other hand model developers tend to specialize models toward a particular application. Thereby models become more complex and may only be applicable under narrow boundary conditions. This leads to a constant re-development of equal models as regards content and objective for slightly different domains. A major drawback of model frameworks is often that knowledge on models can not be represented as desired due to constraints by the chosen modeling language (e.g. mathematical notation) or by the desired model environment. In other words, the translation of the model

functionality into the given modeling language may only be possible to the cost of expressiveness.

Depending on the objective a model evolves through various phases starting from some conceptual level up to a mathematical model in most cases, see figure 2.13. Computer aided modeling requires furthermore the implementation of the model into a particular software environment to be able to use software provided solvers. Assumptions, objectives and boundary conditions may get lost to the end in the process of model development.

It appears that a main problem in model development lies in the lack of appropriate languages to capture knowledge on processes and involved materials and substances (Bieszczad, 2000; Hofmeister, 1998; Lohmann and Marquardt, 1996; Marquardt, 1995b). Another problem comes with in the attempt to formulate general model objectives but oppositely allowing to incorporate specific application oriented aspects (Drengstig *et al.*, 1997; Evenson and Baetz, 1994; Linninger, 2001; Rotstein *et al.*, 1994; Statyukha *et al.*, 2008).

From the above analyses the following key problems on model frameworks can be summarized. Processes and physicochemical or biological phenomena may not sufficiently be formalized by mathematical equations only. However equation oriented approaches are often chosen since model application mostly involves simulation over time (e.g. solve differential equations). There are no meta languages or general modeling languages. In contrast, existing modeling languages are application or software tailored.

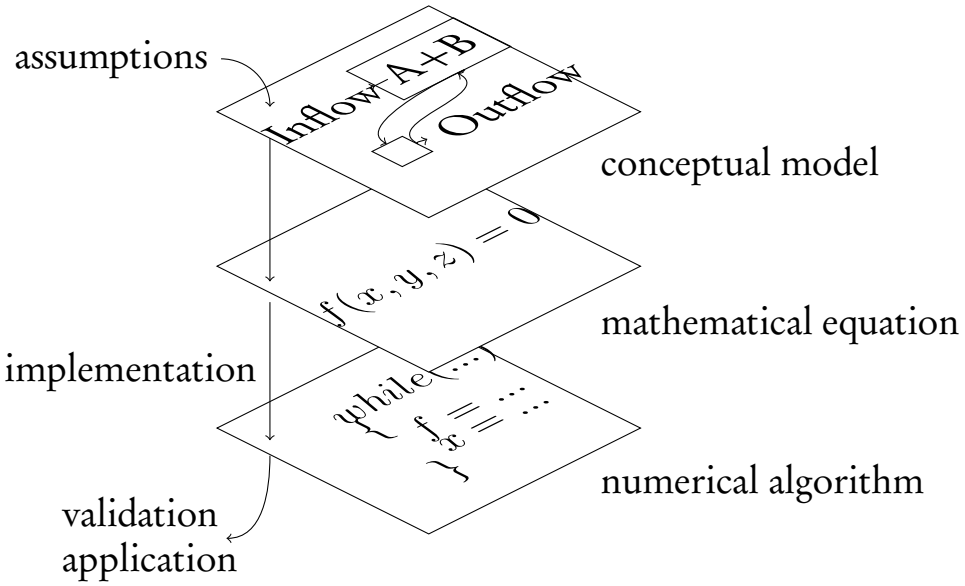


Figure 2.13: Evolution of Process Model Representations (adapted from Bieszczad 2000)

In contrast to the identified drawbacks during model development the following desired features on modeling frameworks can be listed (Braunschweig and Gani 2002, p. 93, Bieszczad 2000, p. 39-47 and Wedel *et al.* 2002, p. 93-94). Modeling frameworks need to allow for declarative model description. This intersects with the requirement to separate knowledge on model functionality from knowledge and how this knowledge is processed (see section 2.4.1 - five principles on knowledge representation). A general declarative model description (application independent description) would furthermore ease the use of models by different model environments. The main requirements on model representation can be summarized as follows.

1. Model representation should allow for hierarchical structuring of model fragments to enable model reuse. The introduction of hierarchies of model fragments would furthermore allow the capturing of different degrees of detail depending on model objective.
2. Model representation should incorporate context specific information (i.e. boundary conditions, model objective etc.)
3. Physicochemical as well as biological phenomena must form the basis for process models.

Besides representing process models there is a need for a concise representation of states (or state variables). Yang *et al.* (2003) summarized this need as follows. Representation of states must comply to fundamental laws of thermodynamics, conservation of mass and energy. Model representation must further include reaction kinetics (e.g. substance specific), as well as transport phenomena.
Further requirements arise in finding the balance between completeness and extendability as well as reputability and perspicuity.

2.5.5 Modeling Process

The term modeling process (syn.: model development) is used for the methodology to develop a model for an arbitrary application. Following the definition of Wedel *et al.* (2002, p. 89): "The purpose of modeling as a work process is to transform the perception of reality or an idea into a symbolic language, which consists of a set of well-defined concepts with an agreed understanding. There are many possible models of a certain perception, but only few of them qualify as being useful to answer the questions relevant in the current modeling context. A modeling process should thus systematically lead to a useful model

and discriminate those irrelevant." Various formalized procedures have been presented to aid the process of model development. One of them, the generic modeling procedure, is suggested by Marquardt (1995b, p. 599-601).

The process starts with the models perception on a phenomena and a modeling objective. It follows the collection of assumptions that merge into a conceptual model. To be able to apply a model by help of a computer program the conceptual model must be translated into a formal unambiguous language which can be understood by a computer (i.e. model implementation). Before the model is eventually applied it must be validated to ensure the correct behavior of the model in contrast to the modeled phenomena. Depending on the validation step assumptions may be rejected or adapted which leads to a loop within the development process. The described development process is schematically shown in figure 2.14. The major steps of the modeling process are described in more detail below. Some more detailed information on model development can also be found in Stein (2008).

Documentation

In parallel to the specious steps of model development building up on each other each step should be well documented. Although this is an often neglected issue it is essential for use and reuse of a developed model. Documentation thereby preserves informal knowledge starting with the initial problem specification, ideas, requirements, modeling assumptions up to the actual purpose of a model. The documentation of the modeling process is similar to design rationale within process design grasping the decisions and assumption which have been taken to develop a system configuration or process structure. In most cases any documentation regarding the modeling process if at all is detached from the developed model itself simply because the used model representation allows no informal representations.

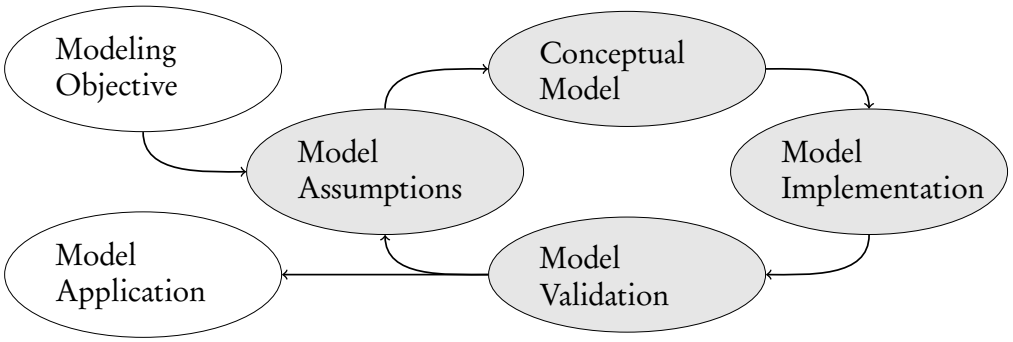


Figure 2.14: Modelling Process (adapted from Wedel *et al.* 2002, p. 90)

Conceptual model

The forming of a conceptual model as intermediate step in the overall development process may not be necessary for simple models but is advisable in most cases since it enables effective communication between domain experts and modelers. The reuse of existing models without the development of a conceptual model lead to errors and inconsistencies in the overall model since the underlying assumptions are not appropriate in the actual modeling context. A conceptual model specifies a model in terms of domain relevant concepts with its assumptions. The type of representation form is oriented on the ability to easily express relevant model assumptions and is therefore independent of a realization in a particular simulator. In some cases there will be not just a single conceptual model but the model evolves through various levels of conceptualization starting from informal to formal notations (e.g the representation in mathematical equations as a formal language may be the last step of a conceptual model before it is implemented). Generally a model must evolve through a series of representations in order to close the gap that exists between a real process and a valid computational model of that process, see figure 2.13.

Implementation

In order to apply a model by a modeling tool on a computer it must be implemented within the particular modeling tool. Thereby engineering concepts must be translated into the concepts provided by the tool in an appropriate and systematic manner. In most cases there are differences between the expressiveness of the conceptual model and a possible implementation within the chosen tool. To minimize the loss of expressiveness in the translation process, conceptional models are designed towards the later implementation in a given modeling tool.

Validation

Each developed model must be validated against the modeled phenomena or system to check whether the chosen modeling assumptions are a useful abstraction of reality. If the model behavior varies significantly from the original system behavior assumptions and the overall model set up must be altered or revised.

Application

As stated before there are various types of model-based applications for example simulation, design or process optimization. There is no modeling tool

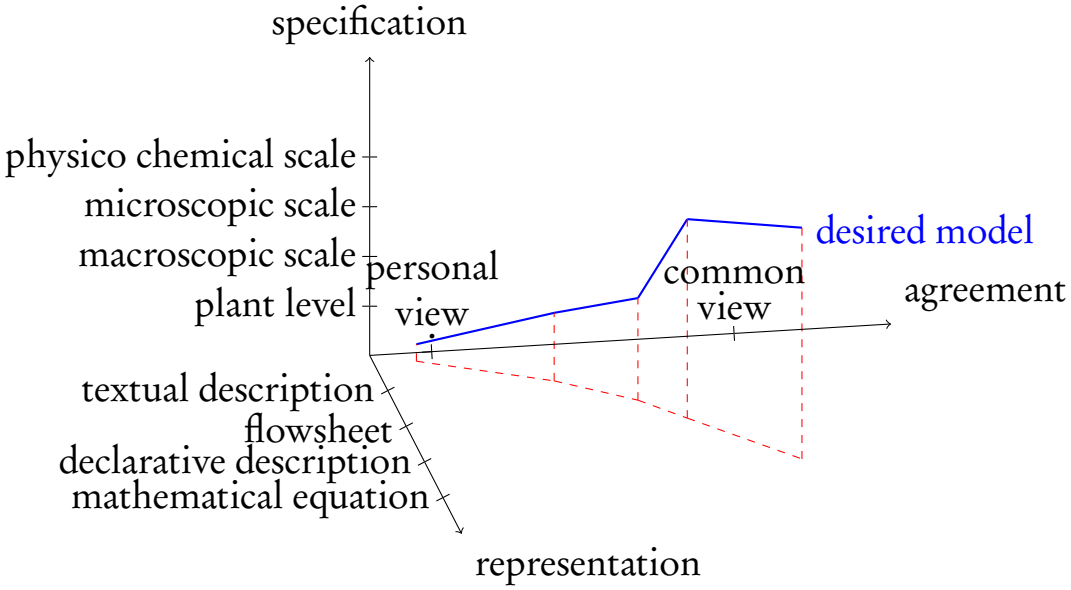


Figure 2.15: Three dimensions of process modelling (Bogusch and Marquardt, 1997)

which offers the full range of model-based applications. Even worse, model implementations of different software packages are generally incompatible.

The above described steps in the modeling process before the actual model application are highly intuitive and are usually performed by a number of domain experts and modelers. The intuitive character and case-specific nature of the modeling process makes it difficult to atomize this process. Nevertheless some parts of the development process may be formalized and can therefore be supported by modeling environments. The following parts may be aided by computers: model creation and manipulation, model analysis, and model translation. More information on computer aided model development is given in Wedel *et al.* (2002, p. 107-112).

In the latter overview on model development it becomes clear that a single model with a defined objectives evolves through a number of intermediate steps. This evolution is schematically depicted in figure 2.15 taking into account the dimensions of agreement, type of representation and specification (model granularity).

2.5.6 Modeling Concepts

2.5.6.1 Introduction

For the process of model development the model framework provides model concepts and a modeling language. Model concepts thereby define how a model is constructed (analogical to model architecture).

In particular model concepts provide the structural, behavioral and material view of a model. Thereby the three perspectives are solely supporting abstractions to filter particular information of a model. In most model representations assertions on structural and material information are not explicitly defined. Instead these assertions are indirectly predefined. In contrast to model concept, modeling languages define how a model is formally represented (see section 2.5.7).

Regarding model concepts chemical engineering models can be broken down to so called fundamental model objects (syn.: canonical modeling objects, building blocks, or simply modeling objects). These fundamental model objects represent core concepts from the application domain (Wedel *et al.*, 2002, p. 108). The idea behind modeling objects are rooted in General Systems Theory (GST) (Klir, 2001; Klir and Elias, 2003) and in so called State-Task-Network (STN) (Gani and Papaeconomou 2006, p. 57; Mangold *et al.* 2002; Marquardt 2005, p. 115-121; Samantray and Bouamama 2008).

According to systems theory, a model is usually regarded as decomposable into a number of subordinate models so that the aggregation hierarchies of arbitrary depth can be developed. Hierarchical decomposition limits the complexity of the modeling effort by focusing on individual parts of the overall model. Furthermore parts of models may be reused (Wedel *et al.*, 2002, p. 97).

At present, the most profound realization of a generic process model which incorporates the ideas of structural, behavioral and material perspectives has been presented in the declarative model for chemical engineering processes OntoCape[6]. A detailed overview on this model is given in Marquardt *et al.* (2010).

2.5.6.2 Structural view

Regarding model structure two conceptual different classes can be identified namely devices and connections. Devices may also be referred to as region objects or shells (Hangos and Cameron, 2001, p. 474). A device represents a delimitable part of a process such as a reactor vessel. Devices themselves are linked alternating by connections. The distinction of connections and devices follows from their role within a process. The role of a device is the deduction of some characterizing state variables (x) such as temperature or concentration from known fluxes (ϕ) such as mass, energy or momentum flow from attached devices (Wasbøand Foss, 1996), see figure 2.16. In other words a device reacts to changes of the corresponding fluxes (a change of a flux is followed by a change

[6] `http://www.ontocape.org`

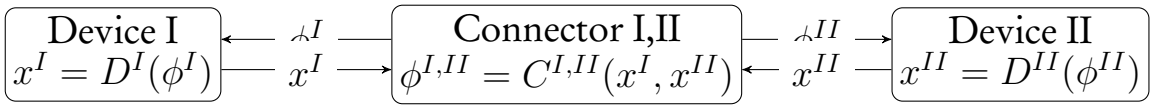


Figure 2.16: Information between devices and connections (Mangold *et al.*, 2002; Marquardt, 1995b)

of a state variable). Oppositely connections determine leaving fluxes depending on the differences of potential size of the adjacent state values. Hence the role of connections is complementary to a device since it reacts to changes of adjacent state variable by changing the corresponding fluxes (Marquardt, 2005, p. 131). Consequently, in dynamic modeling only devices may reflect a holdup for extensive quantities. Devices are therefore described by differential equations with respect to time. For connections no time differences can be taken into account.
Within most applications a clear distinction regarding the structural role may not be found. Instead the role of a particular process model is implied by the knowledge of the user.
Devices and connections can be further distinguished into elementary devices and connections and composite devices and connections. They are termed composite if they form aggregates of devices and connections and oppositely they are termed elementary if they are not decomposable in a certain modeling context. Therefore devices and connections can be used to form structured models of process sub and control units (Marquardt, 1995b, p. 597).

2.5.6.3 Behavioral view

The behavior of modeling objects is reflected by the values of assigned process quantities. A possible taxonomy of process quantities is shown in figure 2.17. Marquardt (1995b) identifies four major groups for the behavioral description of a phase: generalized fluxes, thermodynamic states, state functions and phenomenological coefficients.
Generalized fluxes describe the variation of holdup (change of storage), transport in and across phase boundary as well as sources caused by reaction or some external potential field. The fluxes are depending on thermodynamic state functions and on phenomenological coefficients. Any process quantity assigned to a particular phase may depend on one or several coordinates such as time and spatial dimensions. Any process quantity can be further specified by: name, range, unit, dependency, etc.
The value of each process quantity is restricted by laws. A cutout of a taxonomy of model equations based on physiochemical laws is shown in figure 2.18.

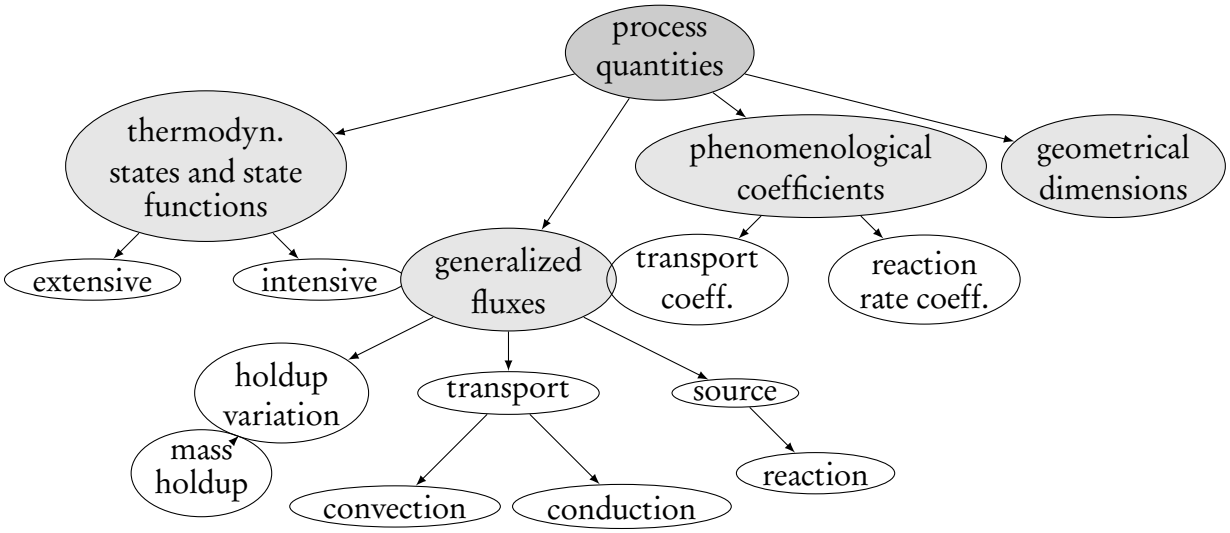


Figure 2.17: A taxonomy of process quantities (adapted from Marquardt 1995b)

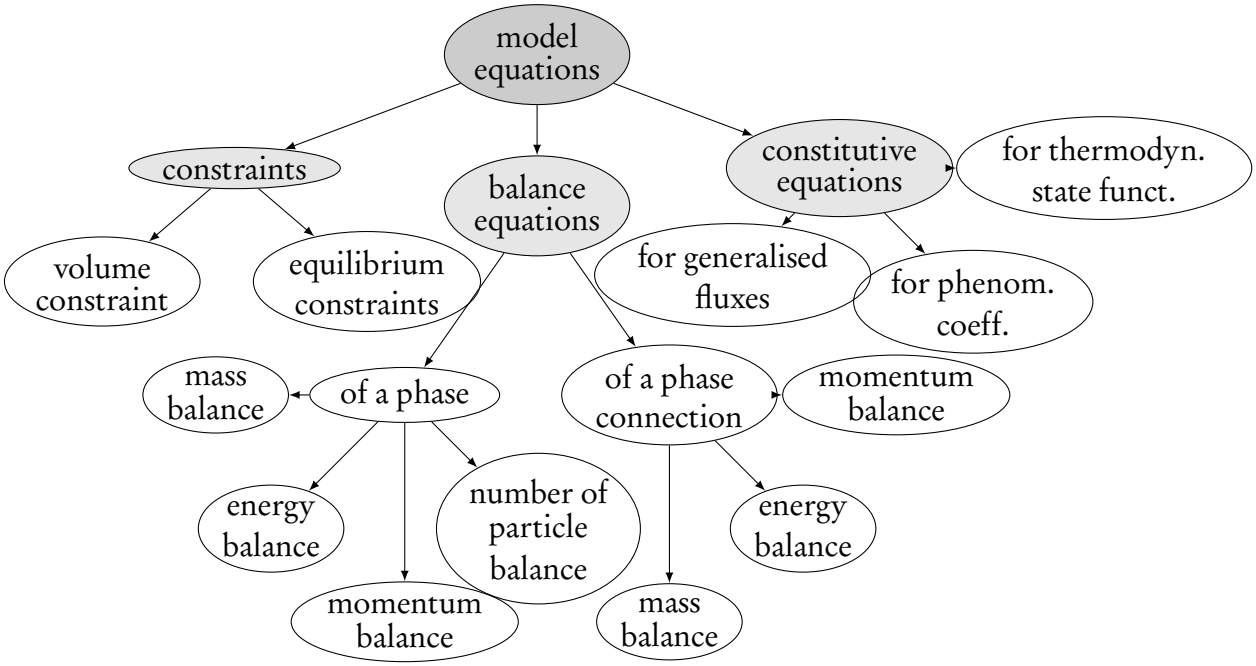


Figure 2.18: A taxonomy of model equations (adapted from Marquardt 1995b)

If a law is derived from fundamental or empirical physiochemical relationships they are referred to as white box models. Oppositely if the are derived solely by experiments the are referred to black box models (Arizmendi-Sanchez and Sharratt, 2008; Marquardt, 1995a).

2.5.6.4 Material view

Whereas most process models contain explicit or implicit assertions on structural and behavioral information material data is often not incorporated. Nevertheless for phenomenological process models in chemical engineering material data is of immanent importance. According to Linninger *et al.*

(1996b): "the material model is an abstract data model for the consistent representation of process streams and their molecular phenomena. Its goal is to capture the thermodynamic behavior of process streams, phases and chemical compounds through a knowledge-based, context sensitive data model. [...] the material model stores the state parameter of process mixtures through its attributes, i.e. pressure, temperature and composition etc. [...] Complex chemical interactions depend on the concentration of the compounds within a mixture, e.g. formation of phases, phase equilibria. etc., [and] can now be interpreted with respect to the actual context."
Following from the definition of unit operation of Arthur D. Little (see section 1.1) the occurring processes within a process unit are a function of the present or provided context within a unit. The objective of assertions of material data is therefore to describe the substances within a process stream in the context of present boundary conditions.
Material in this respect is referred to as all matter of a state with attributed mass and consumed space (Gold *et al.*, 1987). Thereby material provides an abstract description of matter (Morbach *et al.*, 2008d).
The sole description of chemical substances is included in various commercial simulation tools in the form of multi-component databases. Those databases contain a variety of pure-component and mixture property data, parameter regression tools, and services for calculating physical properties and phase equilibria. A survey on those databases is provided by Yang *et al.* (2003). However all the existing databases follow an application oriented approach instead of a unified general view.
Yang *et al.* (2003) proposed a data model on materials that allows for different levels of detail for different development stages starting on a conceptual level. The presented approach merged into the process model ontology summarized in Marquardt *et al.* (2010) and Morbach *et al.* (2008d). The approach of Yang *et al.* (2003) follows the need for consistency to theories of equilibrium and non equilibrium thermodynamic, chemical kinetics and transport phenomena. In the domain of material data in the context of process engineering, Yang *et al.* (2003) identified three distinctive categories to describe material information:

1. Thermo-physical nature of matter characterized by a set of physical constants (e.g. molecular weight, critical properties). Properties of this category are of intrinsic nature.

2. Properties of matter regarding physical contexts.

3. Mathematical models capturing thermodynamic laws or empirical correlations that represent the mathematical relations among the properties of matter.

According to the above categorization material data is modeled by three partial data models, namely substance, phase system, and mathematical model of phase system. The critical concept for distinguishing substance and phase system is the physical context (i.e. temperature, pressure, and composition). Matter without a consideration of the physical context is modeled as substance. A substance is either a mixture or a chemical component. Mixture represents a collection of chemical components. A chemical component, in turn, is defined as a substance that is not intended to be further split when occurring in a certain mixture. A chemical component has chemical component constants representing the intrinsic characteristics of matter such as molecular weight and critical properties.
Analogue to the categorization above Linninger and Stephanopoulos (1998) and Linninger *et al.* (1996b) propose a similar approach. The development of the material model by Linninger and Stephanopoulos (1998) was constricted by the absence of a concise representation language. This also holds for the work of Bieszczad (2000).
Whereas the use of the term of material models originates from the use of process models in chemical engineering it can be argued that the process description of biological carbon removal as presented by Henze *et al.* (1987) can be understood to some extent as a material model as well. Since the ASM approach describes involved substances and component mixtures related to reactions amongst each other as well as to boundary conditions such as temperature, pH value.

2.5.7 Modeling Languages

2.5.7.1 Introduction

In the latter sections modeling concepts have been discussed. Eventually these concepts must be noted in some formal unambiguous modeling language (Wedel *et al.*, 2002, p. 94). Along with the large number of software tools to facilitate computer aided process modeling a variety of languages for model representation have been developed in recent decades. All of these approaches can roughly be differentiated into programming languages, generic modeling languages and domain-oriented modeling languages. Through programming languages models are directly implemented into the solver. Programming lan-

guages offer no specific model representation and hence allow for no model reuse or joined model documentation. Generic modeling languages are mostly based on GST and offer no domain specific concepts. Domain-oriented modeling languages in contrast do offer domain specific concepts (e.g concepts used for model representation in chemical engineering). Besides these complex approaches there is a common use of spreadsheets. This is because spreadsheets (e.g. MS Excel) offer an easy to use interface to underlying procedural programs. Nevertheless they do not offer means of sound model representation.

Languages for model representation vary in terms of generality (i.e. providing general, domain, or application oriented concepts) and in the degree of expressiveness. Depending on this properties modeling languages may support model reuse, model documentation and compatibility between software tools for model application. All languages use some sort of direct or indirect mathematics to represent models.

The following sections provide a more detailed insight into general modeling languages and domain oriented languages.

2.5.7.2 Generic modeling languages

Generic modeling languages (syn.: general modeling languages, or multi-domain modeling languages) do not provide domain specific concepts but concentrate on a mathematical (or systems) level. This branch may be subdivided into mathematical modeling languages, systems modeling languages and representation languages. Thereby rigorous modeling based on physical insight usually apply continuous models regarding state and time (Wedel *et al.*, 2002, p. 94).

Mathematical modeling languages

Using a mathematical modeling languages (syn.: equation based approach) a simple model is represented by a set of variables x and a set of functions f. Thereby variables represent system states, inputs and outputs, the equations constrain possible values of the variables and thereby represent knowledge on the domain of interest. Steady state can be represented by notations as shown

in equation 2.29.

$$f(x) = 0 \tag{2.29}$$

$$f\left(\frac{dx}{dt}, x, t\right) = 0 \tag{2.30}$$

$$x(t) = \int_0^{t_e} \frac{dx}{dt} dt \tag{2.31}$$

The equations f define balance equations describing equilibrium states such as physical and chemical equilibrium or non-equilibrium processes such as reactions. In most cases the equation system may contain more variables than equations necessitating additional specifications for a number of unknown variables in order to achieve a consistent mathematical formulation. Whether the inputs on a formulation are defined or not the model is referred to as open form model or closed form model.
In most cases trajectories are of interest (i.e. transition from one state to another) for example due to a change of inflow into a reactor. This can be represented by differential-algebraic equations, describing the change of balanced quantities with time (e.g. mass, energy, momentum), equation 2.30. The solution of equation 2.30 consists of finding the integral shown in equation 2.31. Since symbolic solutions of 2.30, 2.31 are almost always impossible to find for relevant engineering applications, a numerical integration algorithm has to be pursued, staring from consistent initial conditions.
Most equation-based modeling languages such as ASCEND, OMOLA or gPROMS provide means of abstraction by grouping variables and equations with regard to a certain system into a single object. Inheritance reduces redundant modeling by grouping similar classes under a single parent class and also promotes model reuse by allowing new models to be defined through modification and extension of an existing model class.
The major drawback of this approach is that mathematical formulations are essentially context-free and hence are not linked to chemical engineering concepts. Although the use of a symbolic form (in contrast to procedural form) as equations facilitate model reuse it inhibits an explicit definition of model assumptions. For complex systems a model may grow to hundreds of equations while modifications are made by different modelers. As soon an assumption in the development process is changed or added existing model equations must be analyzed whether modifications become necessary throughout the system of equations.

More detailed information on mathematical model representation can be found in the work of d'Anterroches (2005); Gujer (2008).

Systems modeling languages

Languages of this type use some sort of a system model as a basic concept. Each system model contains equations of type 2.29 to 2.31 and can be connected to other models to form a model structure. It follows therefrom that an output of some model is equal to the input of another model. Model connection is thereby achieved by standardizing interfaces between models. Examples for systems modeling languages are CSSL (Continuous Systems Simulation Language) (Augustin *et al.*, 1967), ACSL (Advanced Continuous Simulation Language) (Mitchell and Gouthier Ass., 1992), Omola (Nilsson, 1993), Dymola (Elmqvist *et al.*, 1996), Modelica Modelica Association (2000), gPROMS (Process Systems Enterprise, 1997). A detailed description on systems modeling languages is presented by Wedel *et al.* (2002).

Representation Languages

Representation languages are used indirectly to represent models since some higher modeling languages (e.g. phenomena-oriented modeling languages) build up on representation languages. Thereby a representation language may serve as a language definition layer where general concepts for process models are constructed which then in turn are applied by a domain-oriented modeling language to formulate a particular process model. Thereby the language definition layer defines the syntax of a modeling language.

Two examples of representation languages which have been used as a language definition layer are VDDL and a context-free grammar for MODEL.LA.

VEDA Data Definition Language (VDDL) has been basically developed from scratch since no prevalent general modeling language was available in the early nineties for the representation of Verfahrentechnisches Datenmodell (VEDA) (section 2.5.7.3). Detailed information on the use and set up of VDDL in VEDA are given by the following authors: Baumeister (2000); Marquardt (1992); Molitor (2000); Reinhardt (1995) and Morbach *et al.* (2008b, p. 95).

Thereby VDDL is a frame-based language that combines features from object-oriented modeling and description logics. Entities are expressed through classes and can be structured by meta- and sub-classes. Class definition is achieved by attributes which can be restricted by facets, methods and laws.

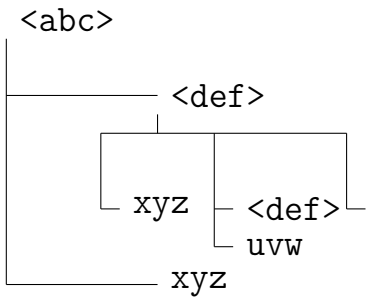


Figure 2.19: Example Production Tree (Bieszczad, 2000)

As second example the formal syntax definition of MODEL.LA (section 2.5.7.3) is based on a context-free grammar (Sipser, 1998). A context-free grammar is a 4-tuple (V, Σ, R, S), where

1. V is a finite set called the *variables*,
2. Σ is a finite set, disjoint from V, called *terminals*,
3. R is a finite set of *production rules*, with each rule being a variable and a string of variables and terminals, and
4. S is the start symbol.

Within the grammar of MODEL.LA variables are depicted as bracketed strings and terminals as `Arial` font text. A production rule than is a variable and a string of variables and terminals separated by the symbol $\rightarrow$. For example, `<abc>`$\rightarrow$`<def>xyz` is a rule that produces `<def>xyz` by substitution with `<abc>`. Alternative substitutions for the same variable can be written in the same rule, with each substitution separated by the pipe symbol |. For example `<def>`$\rightarrow$`xyz|<def>uvw|` is a rule where variable `<def>` may be substituted for one of the three things: `xyz`, `<def>uvw`, or the empty string. Substitution rules can be depicted hierarchically using a tree-like structure, which is referred to as production tree. In this manner figure 2.19 illustrates the two rules described above. The variable on the left hand side forming the first rule appears at the top root of the tree (i.e. `<abc>`). The variables and terminals introduced on the right hand side of each production rule appear as horizontal branches. For those rules with alternative substitutions, each set of possible set of variables and terminals appear on separate vertical branches. Note that the second second substitution (i.e. `<def>uvw`) is recursive.
Besides common engineering approaches for model representation under the general term generic modeling languages, there is a strong emerging field of

representation languages associated to the research areas of knowledge representation, software development and the fast evolution of Internet related technologies. Some important examples are UML, markup languages such as XML, DAML+OIL and OWL. Although these languages are primarily not intended for representation of chemical engineering models their use for this purpose becomes more and more attractive. That is due to their high degree of use and acceptance. It can be expected for the future that languages such as OWL(DL) will play an important role in organizing model frameworks in the area of chemical engineering. An example is already given by the OntoCAPE Ontology presented by Marquardt *et al.* (2010).
Another abstract modeling language, similar to VDDL, has been presented in the work of Linninger *et al.* (1996b, p. 425).

2.5.7.3 Domain-oriented modeling languages

Whereas generic modeling languages focus on the level of general systems representation domain-oriented languages allow model representations to stay close to mental models regarding chemical processes (Wedel *et al.*, 2002, p. 102). In this manner domain experts are able to express knowledge on models more conveniently. Within the process of developing models domain-oriented modeling languages ease the direct representation of conceptual models in a tool, abstracting from mathematical details presented in the former section. Within domain-oriented languages one may roughly differentiate between the use of flow-sheet simulators and the use of phenomena-based modeling languages. Whereas the first is strongly oriented on a graphical representation accompanied by a strong dependency on the actual modeling tool phenomena-oriented modeling languages focus on a profound representation of models based on its underlying physical or chemical phenomena.

Flow-sheet simulators

This approach is also referred to as *block-oriented* or *sequential-modular approach* (Bieszczad, 2000, p. 26). The concept of unit operations has already a long history and originates on the work of Arthur Little at MIT in 1915. The structure of a conceptual model is thereby described by balanced volumes and connecting streams. This is close to mental models of a reactor which might explain the success of this approach. Consequently this approach is implemented in most commercial process modeling tools today. Some examples are ASPEN PLUS by Aspentech, HYSIS.PLANT by Hyprotech, or PRO II by Simsci (Braunschweig and Gani, 2002, p. 102). A distinct attribute of flow-

sheet simulators is the presence of a library of precoded (predefined) models with the granularity of unit operation. Every process is abstracted by a block diagram consisting of standardized blocks (i.e. process units). Blocks themselves are linked by signal like connections. These connections facilitate the flow of information, material, or energy. The behavior of each balance unit or building block must be defined either by equilibrium or non-equilibrium phenomena. In most cases it is not possible to alter the underlying phenomena and incorporated domain knowledge. A particular process configuration is simply achieved by choosing available building blocks and to connect them. The major drawback of this approach is in most cases a fixed level of model granularity. Inevitably this approach results in a strong dependency of model representation and applied software tool.

Phenomena-based modeling languages

This approach is also referred to as process modeling or process oriented languages. Oppositely to flow-sheet simulators phenomena-oriented approaches are almost only elaborated by a view academic research groups so far. Relevant literature in this regard is summarized in the following publications Braunschweig and Gani (2002, p. 103), Arizmendi-Sánchez and Sharratt (2005); Linninger (2001); Linninger *et al.* (2000b); Rodriguez (2005) and Bieszczad (2000). The development of phenomena-oriented languages is strongly driven by the requirements on model frameworks presented in section 2.5.4. Furthermore phenomenological modeling languages build up on generic modeling languages. The main requirements and development criteria are:

- to provide various levels of granularity (multilevel modeling) ranging from single phenomena to unit operations
- to separate declarative from procedural knowledge
- to separate structure description from behavior description
- to achieve a clear separation between model language and modeling environment
- to support the hierarchical decomposition approach
- to allow for representation of process models through the description of elementary physical and chemical phenomena

Two examples for phenomena-oriented languages are MODEL.LA and VEDA. MODEL.LA introduced by Stepanopoulos *et al.* (1990a,b) was the first modeling language which implemented the ideas of a phenomenological modeling approach. Bieszczad (2000) further developed the framework MODEL.LA. MODEL.LA provides a library of concepts from chemical engineering sciences such as conservation principles, equilibria, reaction kinetics, transport mechanisms etc. The language itself is composed by a set of modeling elements and semantic relationships. Modeling elements thereby represent concepts as system, fluxes, reactions, materials, etc. Whereas the latter describe how modeling elements interrelate within a particular model. To formally define the syntax of the modeling language a context-free grammar by a 4-tuple is applied. Following the principle to strictly separate declarative model specifications from procedural modeling knowledge regarding the modeling activity the latter is captured by so called modeling logic. The applied framework of modeling logic allows for the independent description of modeling knowledge apart from implementation. Modeling logic consists of a set of logical operators (e.g. 'if-then' statements of knowledge). These logical operators are defined in terms of modeling elements and semantic relationships by the modeling language, see figure 2.20. The modeling language and the logical framework have been integrated in a computer-aided modeling environment. Through a graphical interface the modeling environment assists users in the process of model development by automatically deriving requisite model equations.
Applying the grammar of MODEL.LA all necessary concepts are formalized and are rooted in the first production rule which is shown below (Bieszczad, 2000, p. 25):

```
<phenomena-based model>  →  <structural characterization>
                            <chemical characterization>
                            <derivation context>
```

The second example of phenomena-based modeling language in this context is VEDA. VEDA has first been published by Marquardt (1992, 1995b). It uses the object-oriented approach for conceptualization instead of predicate logic calculus (Baumeister, 2000; Molitor, 2000). Formal representation is achieved by the formal representation language VDDL. VDLL defines frames for modeling concepts and modeling relations by tuples of attributes, laws, and methods. The major modeling concepts implemented in VEDA are depicted in figure 2.21. VEDA is applied in the modeling environment MODKID (Bogusch *et al.*, 2001).

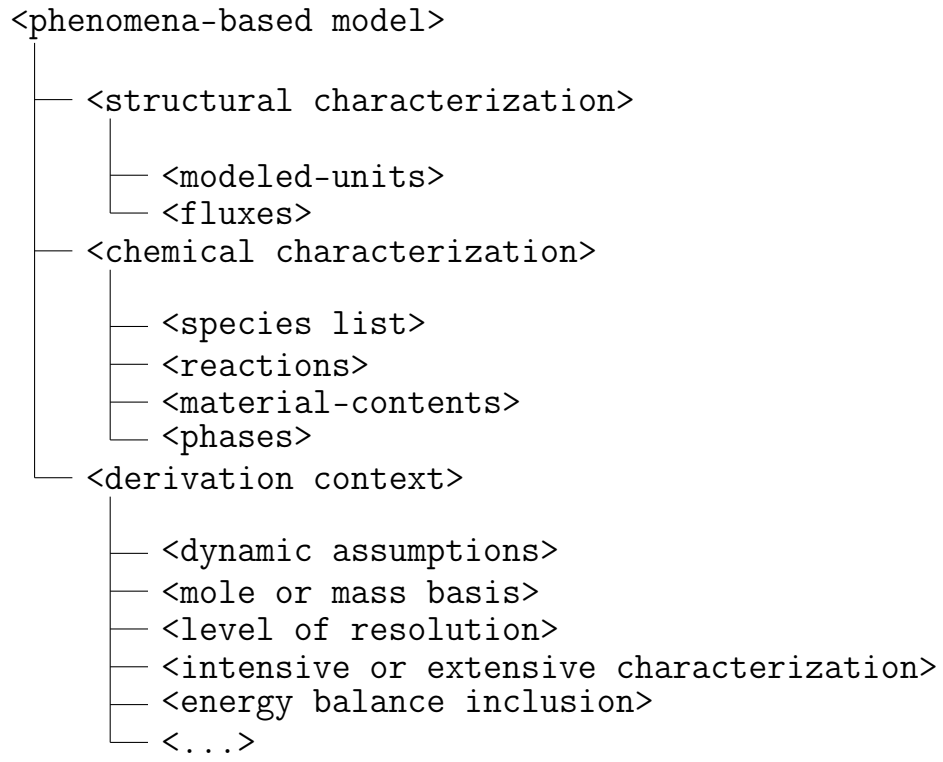


Figure 2.20: Expanded Phenomena-Based Model Production Tree (Bieszczad, 2000)

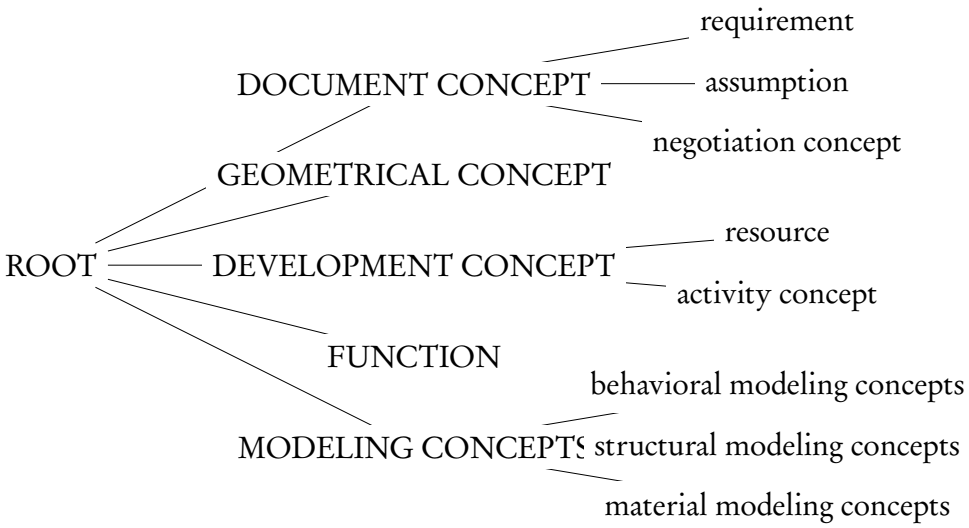


Figure 2.21: Major concepts in VEDA (Bogusch *et al.*, 2001)

2.6 Process Design and Optimization

2.6.1 Introduction

In the former section on process modeling fundamental concepts and languages used for model representation have been reviewed. In this section the focus is on how this knowledge on processes and materials can be used for computer aided process design. The pure objective of process design lies in the

definition of the AI term planning that is to find a sequence of actions to reach a defined goal (Russel and Norvig, 2003, p. 465). In short, a plan is a sequence of actions to reach a desired objective (MCDermott and Hendler, 1995, p. 3).

A general definition of the term design is given by Li (2004) as the systematic, intelligent generation and evaluation of specifications for artifact's whose form and function achieve stated objectives and satisfy specified constraints.

Within the field of chemical engineering the design of processes is often referred to as process synthesis. According to Siirola and Rudd (1971, p. 353) "process synthesis begins by the discovery in the laboratory of sequences of chemical reactions which link readily available raw materials to more valuable products, and it ends with the development of the flow sheet for the commercial process which exploits the chemistry most efficient." Similar the term conceptual design of chemical processes is defined by Douglas (1995, p. 2522) as the attempt to identify all of the decisions required to develop a complete process flow-sheet. Westerber (1989) states that process synthesis is about the automatic generation of design alternatives and the selection of the better ones based on incomplete knowledge. Hocking *et al.* (2002) state that a design problem exists when the desired end result is known, but the means or process leading to the end is not. They furthermore claim that the overall design process passes sequentially through the phases of conceptual design, process design, detailed design and cost estimation. Wherein conceptual design the major process units are selected and possible objectives are screened. During this phase steady-stated simulations are elaborated. During process design process units are looked at in more detail to determine particular operating conditions possibly applying dynamic simulations. In the third phase, detailed design, complete construction diagrams are developed. During cost estimation the overall costs such as construction and running costs are evaluated.

Within this work the scope lies particular on the terms of computer aided process design, conceptual design and flow-sheet synthesis (or flow-sheet generation). Hence the focus entangles methods on how to use existing model representations to automatically generate process chains (or treatment trains) that achieve a defined goal.

In chemical engineering there are various objectives for conceptual process design. The objective might be to optimize existing processes (retrofit) or to develop new ones. In general the objective is to find process chains for the production of end products such as pharmaceuticals, food additives or any other chemical and biochemical products from raw materials or other intermediate

products. In this context the term product design is sometimes used. Last but not least an often set objective is the purification of streams from pollutants and contaminants according to environmental standards and limits. In most cases the evaluation of process designs will be determined regarding the efficiency of cost, energy or use of materials.

The objective of process design also depends on the scale of the process in vision ranging from the development of process on molecular scale up to the development of process chains on plant scale or above. Li and Kraslawski (2004, p. 593) differentiated process design on three scales: meso-, macro-, and micro-scale.

On macro-scale process design is basically concerned with environmental aspects. Due to the complex nature of environmental effects regarding temporal and spatial scales the assessment of environmental impacts has become a multi-objective decision task. The main tasks of process design on macro scale involve the development of indicators for the quantitative evaluation of environmental impacts and issues related to the multi-objective optimization of the process under consideration.

According to Li and Kraslawski (2004) notable research activities on macro-scale process synthesis started in the nineties. In contrast first works on the meso-scale go back to the sixties. The reason for that might be that process design on this level focuses on unit operations which is close to mind models of process structuring in general. The main evaluation criteria hereby is cost-efficiency.

In recent years attention has been directed to process design on micro-scale level. At this scale insight in phenomena at the microscopic or even the molecular level is applied for the development of intensified processes and consumer-tailored products. According to Stankiewicz (2002) the target of process intensification is to make a quantum leap in process and plant efficiency with respect to space, time, energy, raw materials, environment, etc. It comprises novel types of equipment as well as novel processing techniques and process-plant development methods. In a short term, process intensification leads to a substantially smaller, cleaner, and more energy-efficient technology.

Following the overall objective of this work the focus on conceptual process design lies excursively on the meso-scale.

Besides the differentiation according to scale, process design also depends on the process configuration as batch configuration or continuously working system.

2.6.2 Planning Theory

2.6.2.1 Problem Solving

Within AI planning is enclosed into the area of problem solving. In most cases planning can be understood as a systematic search. Thereby problem solving is the process of generating solutions by search. The problem itself is specified by a set of goals, a set of objects and a set of operations (or tasks, actions). Hence as stated in the introduction a plan is a sequence of actions through problem space that connects the initial state to the goal state.

The problem space encompasses all valid states that can be generated by an application of any combinations of operators on any combination of objects. There may be more than one solution contained within a problem space (Chakraborty, 2010).

The complexity of planning results from searching the problem space. If the worst comes to the worst problem space and number of options are related exponentially (combinatorial explosion). The overall planning effort also depends on the per node cost. That is the time or calculation effort on each node to generate a successor node, to determine the goal achievement or due to calculation of heuristics. The main objective in finding an appropriate planning algorithm is to find the optimum between doing more search and doing more work on each node. In short, the task is to limit the combinatorial explosion (Crooks and Macchietto, 1992). Furthermore, the possibility to solve a planning problem is tightly bound to the way a planning problem is represented. The key is to chose a representation that allows the planning algorithm to seize the overall problem structure in the attempt to unfold a complete plan (Russel and Norvig, 2003, p. 468).

The framework for representing a problem space is a directed graph. For simple search algorithms the nodes are connected to only one predecessor node resulting in a tree (hierarchical order). For complex search algorithms cyclic graphs may be chosen or more than one predecessor nodes may be allowed. In classical planning the problem space is fully observable, deterministic, static, discrete and finite.

A general problem formulation includes the representation of states (also impossible ones), goals and possible actions. The states from where the planning problem starts are referred to as initial states. The representation of actions must include a definition of preconditions and effects. Preconditions are a conjunction of literals that specify the boundary conditions that must hold

true for the use of a particular action. Complementary effects describe how a state changes by the use of a particular action.
Search based planning algorithms can be differentiated into state space search and plan space search. Within state space search the problem space is defined as the set of all states reachable from the initial state. Thereby nodes represent states and arcs (syn.: edges) represent actions. A plan is a sequence of states through state space that connects initial and goal state. In state space search the search may start at the initial state (progressive planning) or start from the goal state (backward or regression planning) (Russel and Norvig, 2003, p. 475). In progressive planning one has to deal with a huge branching factor due to irrelevant options. There are two search mechanisms linear and non-linear planning. Linear search algorithm add successively consistent steps (one at a time). This results in an incremental plan evolution. The present system state can be executed immediately. These plans are referred to strictly ordered and fully instantiated. Non-linear planners target towards simultaneous satisfaction of several goals. Opportunistic steps can enter the plan even when preconditions are violated, therefore additional objectives must be posted to regain consistency (the planner posts a sub goal for each violated precondition). This is referred to as goal posting. Steps in a plan are ordered relatively to each other. State changes can only be predicted through projection of the evolving plan. These plans are referred to as non-linear partially-ordered. Linear and non linear planning is sometimes referred to as monotonic and non-monotonic planning (Ali, 1999).
State space search approaches are totally ordered (syn.: strictly ordered planning). This means that all actions are linked to the initial or goal state.
In plan space search the problem space is a set of partial plans. This allows for dividing the overall problem into subproblems that can be solved independently. This approach is also referred to as partially ordered planning (POP) (Russel and Norvig, 2003, p. 479).

2.6.2.2 Uniformed and Informed Search

Uninformed search strategies (syn.: blind search) explore a given search space entirely to reach a specified goal without using further information. This approach is in most cases inefficient regarding time and memory. Informed search strategies incorporate problem specific knowledge to find possible solutions more efficient. Informed search algorithms are the basis of effective problem solving or search based planning. Problem specific knowledge thereby is embedded in the term heuristic. Although uninformed search algorithms may

take forever to find a solution, they guarantee to find the best solution possible (if there is one at all). This may in contrast not be the case for all informed search algorithms. They may be more efficient to the cost of not finding the optimal solution. A heuristic or better a heuristic function $h(n)$ defines the estimated cost for the node with the lowest cost to the goal node (Russel and Norvig, 2003, p. 132). To find for example the shortest path from one spatial point to another spatial point over an arbitrary number of intermediate points the distance between start and end node could be used for the heuristic function (i.e. beeline). The linked list of nodes with a total length close to the direct connection can be evaluated as best solution. Some search methods that apply an heuristic function are greedy best-first search, A* search. The problem of informed searches is that they might get caught in a local minimum (or local maximum). Search algorithm that attempt to overcome this problem are the hill-climbing-search, simulated annealing or generic algorithms (GA). A profound background on uninformed and informed search approaches is given in Russel and Norvig (2003, chapter 3 and 4).

2.6.2.3 Constraint satisfaction problems

A subtype of search based problems are Constraint Satisfaction Problems (CSPs). Informed search algorithms use a heuristic function (estimate the difference between actual state and goal state to choose and evaluate the next step). In other words they use problem specific knowledge. CSP is about search algorithm which use the general structure of states to define generic heuristic functions (Russel and Norvig, 2003, p. 184). In contrast to planning as described above there is a defined set of variables X_i (list of nodes within a constraint graph). Additionally there is a defined set of constraints C_i. To every variable X_i there may be a value assigned out of a non empty domain D_i. Every constraint obtains a subset of allowed values from D. The state is determined by the assignment of values for some or all variables within the constraint graph. Every assignment that does not violate any of the given constraints is referred to as a consistent assignment. In the case of a consistent assignment for all variables a solution is found (complete assignment) (Russel and Norvig, 2003, p. 184).
Within CSP the initial state is defined by an empty assignment (variables with no assigned value). In each step a value is assigned to an empty variable as long as no constraint is violated. After each step goal achievement is tested.
CSPs can be classified according the type of variables and constraints. The used domain of possible values for the variables may be finite or infinite. The

associated values may be discrete or continuous. An example for discrete values of a finite domain is the pair of values true, false. Integer values are discrete but are confined to an infinite domain. Certainly most common constraint satisfaction problems are those with continuous variables. They are solved by linear and nonlinear programming methods.
Regarding the used constraints there are unary, binary and higher order constraints. The simplest form is a unary constraint requiring a precise variable value relation. Binary constraints relate two variables (e.g. $X_1 = X_2$). Higher order constraints relate more than two variables.

2.6.3 Conceptual Process Synthesis Methods

2.6.3.1 Introduction

In the context of conceptual process design on meso-scale there are generally two main branches namely optimization- and knowledge bases approaches. Nevertheless the two branches can to large extend be regarded as complimentary. In optimization based approaches the formulation of the design task takes the form of an optimization problem. Optimization based approaches requires a representation of a superstructure that includes all possible treatment options and connections between them. Associated methods require a given superstructure to be optimized where the optimal solution must be included. knowledge-based approaches focus on the representation and knowledge organization of the design problem itself (Li and Kraslawski, 2004).

2.6.3.2 Optimization based approaches

Optimization-based approaches are similar to CSPs (cf. section 2.6.2.3). Thereby the optimization-based approach focused on a formal, mathematical representation of the problem and optimization (Li and Kraslawski, 2004). But this approach should not be applied under defined problems resulting from multi-objective problems.
The advantage of this approach is the ability to deal with various synthesis problems through provision of a systematic framework. The disadvantage lies in the need of a given superstructure that relates all possible connections. The optimization then reveals the best relations according to some objective function. However the identification of such superstructure is usually identified only by huge computation effort.
From a mathematical point of view optimization problems can be categorized as Mixed Integer Linear Problems (MILP) and Mixed Integer Non-Linear Problems (MINLP). Thereby can MINLPs be solved through stochastic ap-

proaches such as *simulated annealing* and *evolutionary algorithms* (e.g. Generic Algorithm, GA).
Analogue to CSPs for continuous domains applying linear programming the solution time is equal to the polynomial number of variables (Russel and Norvig, 2003, p. 186).
More detailed information on the use of MILP and MINLP for the generation of water treatment related problems can be found in Henüen (2004), Boyle and Baetz (1998), Roda *et al.* (2002), Linninger *et al.* (1995), Eggersmann *et al.* (2002, p. 338).

2.6.3.3 Knowledge-based approaches

Not all knowledge based approaches fall in the category of AI based approaches. For instance the heuristic approach is more a structured methodology in contrast to a formalized representation.

Heuristic approach

The heuristic approach was first introduced by Siirola and Rudd (1971) as a systematic methodology for the synthesis of multi component separation sequences. The methods can be summarized by asking the right question at the right time. In contrast to later developed approaches (solve by problem structure) was the problem representation of lower importance. Based on the work of Siirola and Rudd (1971), Douglas (1985) proposed a hierarchical, heuristic procedure for the design of chemical processes. Thereby heuristic rules are applied at different design levels to generate flow-sheet alternatives. This approach has been implemented in various applications. An exemplary use of the methodology is given by Douglas (1988). It can be concluded that this solution methods brings about the quick location of near optimum solutions. Nevertheless through the sequential design process no interactions between different design levels can be taken into account. Furthermore, the pursuit of multi objective issues is difficult to deal with. In the work of Freitas *et al.* (2000) the heuristic approach is also referred to as structured approach.

Means-Ends Analysis

This approach descends nicely from the AI technique to solve a planning problem by structure. Given an initial and goal state (S^0, S^e) within a state task network this methods sequences all necessary tasks to minimize the difference between initial and goal state (convert S^0 into S^e). If a plan p consists of an uninterrupted chain of steps p, the plan p is referred to as complete plan

(Linninger and Chakraborty, 1999). The intermediate selection of alternatives (tasks) may be supported by the specification of pre- and postconditions. Thereby preconditions confine boundary conditions under which a particular task may be applied. Postconditions postulate the expected effect if a particular task is applied. Nevertheless where a property difference (or state difference) is discovered, it is possible that a selected task may not completely eliminate this difference. Which in turn requires a follow up task for the same state difference. Another side effect of minimizing the difference between states is the decrease or increase of differences of other properties (pareto optimum). If not all properties can be accounted they may be ranked as in the exemplary hierarchy: identity, amount, concentration, temperature, pressure, form.

Applying this approach does not guarantee the design of the best alternative. An exemplary application of the MEA approach is also described in Ali (1999).

Phenomena-driven design

Analogue to the use of phenomena oriented modeling languages (cf. section 2.5.7.3) this approach focuses on the level of phenomena (low level aggregation) rather than at the level of building blocks (Li and Kraslawski, 2004, p. 592). This is in line with the definition from Tanskanen *et al.* (1995) on process design being the control of physicochemical phenomena for a purpose. This methods offers a systematic way to generate a list of desired phenomena and favorable conditions in order to be implemented. Upscaling to the level of building blocks is achieved by grouping favorable phenomena depending on the occurring conditions within a particular building block. Jaksland *et al.* (1995) applied this approach for separation process design and synthesis based on thermodynamic phenomena. Without the restriction of building blocks this method offers the development of innovative units and processes for supporting creative design (Li and Kraslawski, 2004).

Case-base reasoning (CBR)

The key of this approach is to solve new problems by reusing previously applied solutions (Hamouda *et al.*, 2009, p. 1762). This is usually accomplished by a cyclic procedure including at least two steps. The first step deals with retrieving known solutions by a given problem. In other words a new problem is matched against old problems. In a possible second step the problem is altered if no old solution has been found. The advantage of this approach is the possible quick retrieval of solutions. In contrast new design is always based

on old solutions (if there are any at all.). Furthermore this approach requires a profound data base of applicable solutions (Seuranen, 2006).
The CBR approach has been used for instance to solve problems on WWTP operation as described in the work of Ceccaroni (2000, 2001).
Further knowledge-based approaches on conceptual design are Conflict-based approach (CBA), Driving force method, and Axiomatic design. For more information on these approaches see Li and Kraslawski (2004).

2.6.4 Applications for design of wastewater treatment chains

Conceptual process design in the field of wastewater purification targets on the design of flow-sheets for wastewater minimization (water reuse) and the processing of waste streams to meet environmental limits (Alva-Argáez *et al.*, 1998; Bagajewicz, 2000). Hostrup *et al.* (1999) gives the following problem statement: "Given the identity of a chemical species that must be removed from a mixture, determine the optimal flow-sheet with respect to separation efficiency, cost of energy consumption and compounds involved, and process/environmental constraints." More general plant-wide waste management not only targets on the purification and reuse of water but also on the recovery of raw materials (Chakraborty and Linninger, 2002).
As stated before the fundamental difference in contrast to conventional process design in chemical engineering lies in the definition of compounds (in particular organic pollutants) and related processes. Conceptual process design for wastewater treatment applications is also referred to the synthesis of treatment trains.
In this section a number of examples on process design methodologies for wastewater treatment and reuse are examined. Key questions are thereby the used representation of treatment options, pollutants and related processes. Analogue to conceptual process design in chemical engineering the applied methods are parted into those using an optimization approach and those using a knowledge-based approach. Extended reviews on process design for wastewater treatment applications are presented by Bagajewicz (2000); Hamouda *et al.* (2009).
In the works of Ashley and Linke (2004); Linke and Kokossis (2003); Rigopoulos and Linke (2002) a design and optimization approach for wastewater treatment schemes has been presented. In particular the approach has been used for the optimal design of the activated sludge process. As common for most optimization approaches their approach is divided into a setup of a superstructure including all generic synthesis units with interconnecting streams and eventu-

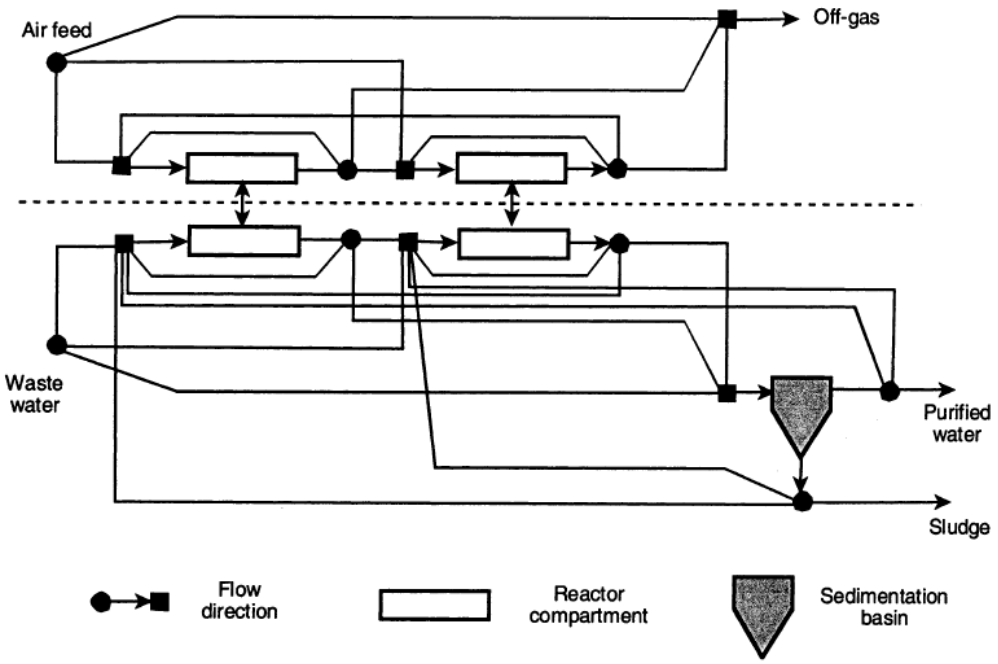


Figure 2.22: Superstructure network featuring two oxic/anoxic reaction units and one sedimenation unit (Rigopoulos and Linke, 2002)

ally an optimization framework to reveal the optimal configuration. The chosen representation of involved compounds and processes is based on the work of Henze *et al.* (1987). The basic elements of the superstructure synthesis generation is based on generic reactor/mass exchanger units (RMX) and separation task units (STU). Having the objective to optimize the activated sludge process for elimination of COD and N_{total} two to three reactors have been included into the superstructure. The chosen number of reactors has been identified as balance to still be able to account for conventional sequences of aerobic-anoxic reactors on one side and to bare the computational effort. Figure 2.22 shows an exemplary superstructure build on two reactor units. The applied objective function can be summarized as to maximize cost effectiveness and minimize environmental impact (Linke and Kokossis, 2003).

The superstructure optimization is tailored to the objective function J given in equation 2.32. Due to the non-linearities introduced by the processes of the Activated Sludge Model No.1 the conventional MINLP method could not be used. Instead a stochastic search technique has been applied (Rigopoulos and Linke, 2002). Thereby does the search allows the moving to dis-improving states, to enable the system to escape from locally optimal states. As a result optimal configurations can be computed depending on the wastewater char-

acteristics.

$$J = min\left[\left(COD + \frac{COD_0}{N_0}N\right) \cdot 100\right] \tag{2.32}$$

Other optimization approaches regarding the development of water and wastewater networks are presented by Alva-Argáez *et al.* (1998); Statyukha *et al.* (2008) to name only a few.
In contrast to the optimization based approach mentioned above Freitas *et al.* (2000) presented an approach for synthesis of wastewater treatment flow-sheets as heuristic approach.
This approach consist of a stepwise design of the different levels of a conventional wastewater treatment chain, that is pre-primary treatment, primary treatment, secondary treatment and tertiary treatment. For each of these four levels they propose different heuristic rules to identify optimal treatment steps. In that sense the proposed approach of Freitas *et al.* (2000) can be seen as a knowledge-based expert system. The included KB covers thereby a set of rules to describe expert knowledge. The optimal task or treatment actions are then derived by a decision tree. The KB contains furthermore information on physical and chemical properties on pollutants and their treat-ability data. The data on pollution properties is taken from Hansen *et al.* (1988).
An exemplary rule set for chromium precipitation from the work of Freitas *et al.* (2000) is given below:

> **if** secondary treatment is biological under aerobic conditions **and** influent concentration on chromium exceeds 2 mg/l, **then** precipitate chromium;
>
> **if** precipitation is used to remove chromium **and** pH is below 7 or pH is above 9 **then** include a neutralization reactor before chromium precipitation;
>
> **if** the treatment includes oxidation **and** includes a precipitation **then** precipitation must follow the oxidation in the flow-sheet sequence;

Similar approaches as the one presented by Freitas *et al.* (2000) can be found in Butner (1999); Flores Alsina (2008).
Further approaches that are solely based on qualitative approaches in the sense that they use no calculations but only rule bases are presented in the works of Boyle and Baetz (1998); Roda *et al.* (2000a,b, 2002); Wukovits *et al.* (2003).

Linninger and Chakraborty (1999) presented a planning methodology for treatment selection and optimization applying the means ends approach. The root of this approach is the assessment of batch process for pharmaceutical products incorporating ecological considerations. The algorithm has been implemented into the "BatchDesing-Kit" (BDK).

In contrast to hierarchical and strict numerical optimization approaches the approach by Linninger and Chakraborty (1999) is founded on a strict problem representation incorporating a phenomenological model description (cf. section 2.5.7.3). A single thermodynamic state of a waste stream is thereby described by its context namely by parameters such as amount m, temperature T, pressures P and composition vector X. For example, the initial state S^0 be denoted by $S^0 = (m, T, P, X)$. Complementary to the states a set of environmental targets define limit values on state properties. Finally goals represent violations on environmental targets depending on state conditions. The final state S^e is characterized by the absence of any rule violations. The objective of the goal driven search (planning) algorithm is to find a sequence of operations to link initial and goal state. The applied approach is therefore referred to as goal-driven. The goal achievement thereby must fulfill the criteria of consistency and opportunity. Consistency is achieved if no preconditions of a step are violated. Opportunity is achieved by choosing a treatment step which produces a successor state which is closer to the desired final state. According to planning theory the applied methodology is a strictly ordered and fully instantiated planner (Chapman, 1987).

Remarkable to other approaches is the application of a material-based design paradigm as basis for state representations (Linninger *et al.*, 1996b). Thereby the material-design paradigm defines the basic moves in the composition space (cf. section 2.5.6.4).

The applied control structure of the planning algorithm is depicted in figure 2.23. The control structure manages the exploration of the problem space selecting options out of those that fulfill defined pre- and post-conditions. Each violation on a set of environmental or legal targets is a goal. In turn the final state has no such violations. Side effects caused by qualitative and quantitative state changes can only be computed by step execution (hampers the use of non-linear planners).

Each treatment option is described by key attributes such as limitations (preconditions), efficiency, postconditions and cost (Linninger *et al.*, 1996a, p. 1434). In contrast to a simple linkage between treatment option and com-

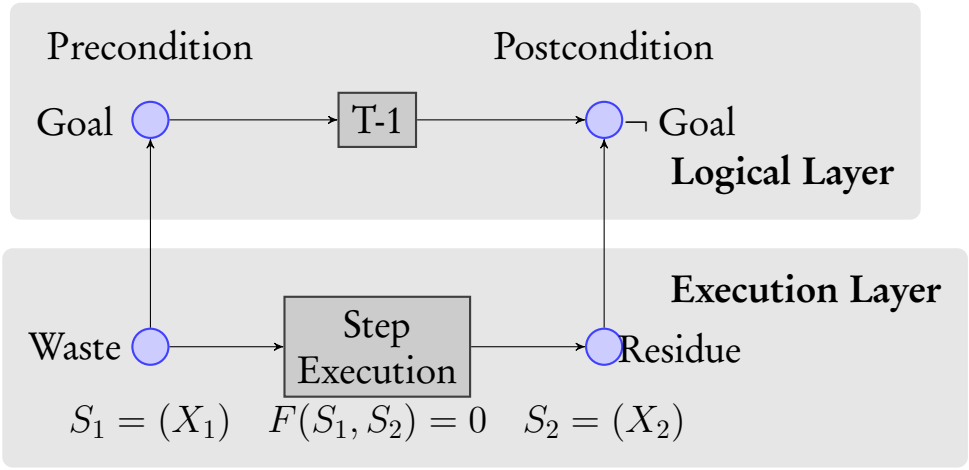


Figure 2.23: Logical layer and execution layer (adapted from Linninger and Chakraborty 1999)

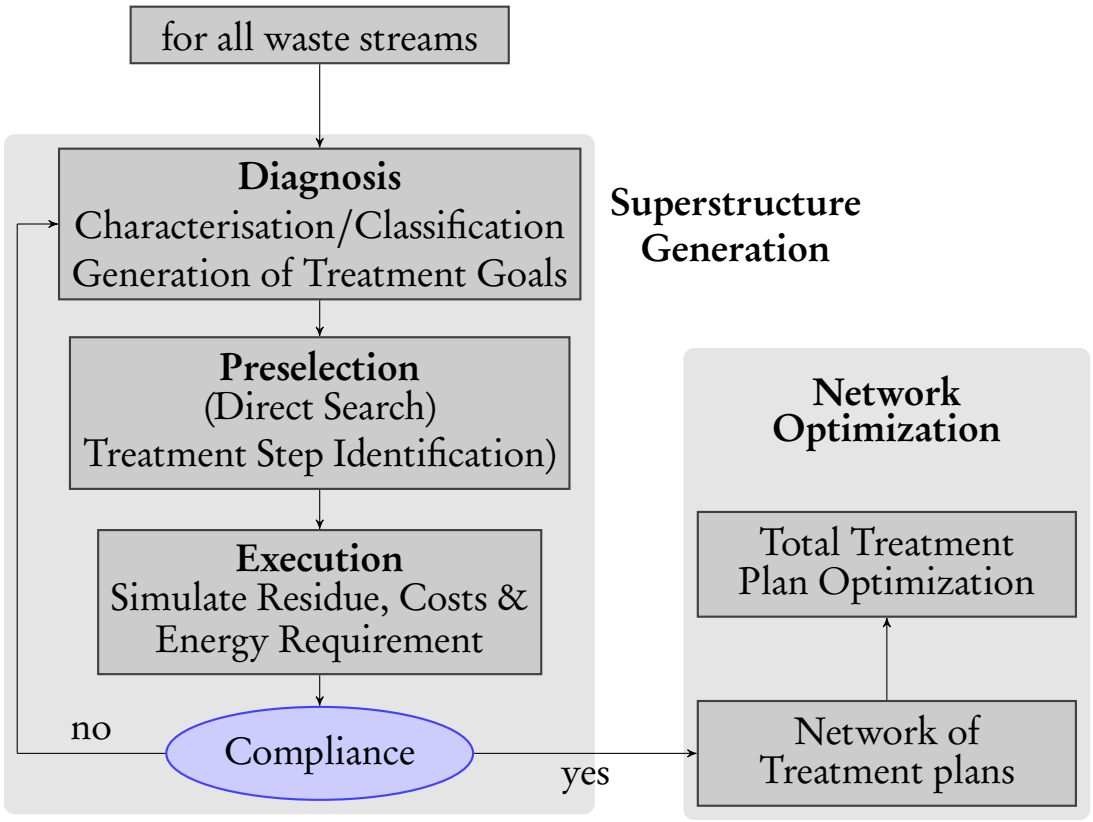


Figure 2.24: Methodology for waste treatment synthesis (adapted from Linninger and Chakraborty 1999)

pound in the manner "$compound_i$ treatable-though $treatment_i$" a context specific treatment selection has been used. Therefore the applicability of a treatment option can be specified to groups of chemicals which share a common behavior and properties.

The overall algorithm to identify feasible treatment schemes by Linninger and Chakraborty (1999) is shown in figure 2.24.

Regarding the use of MEA for generation of treatment chains for wastewater treatment as described in Linninger and Chakraborty (1999) the following drawbacks can be concluded.
The effect of an action depends on the state (unbound variables) which in turn requires fully instantiated planning (forward chaining). After the waste stream becomes diverged the overall problem becomes a multi objective planning. Furthermore MEA cannot handle cyclic operations and is not capable of multi objective problems (Crooks and Macchietto, 1992).

2.7 SUMMARY

Main aspects of the literature review are summarized within this section. Furthermore, this section lists major conclusions that have been incorporated into the model development and application in the context of this work.

In section 2.2 on wastewater characterization a number of quality parameters have been presented. Although no universal classification system for water quality parameters exists, they can be grouped according to unique parameter types (e.g. composite parameters, solids parameters etc.). Since treatment technologies for wastewater treatment are tailored to properties of wastewater constituents (e.g. particle size, biodegradability, etc.) model representation of water quality parameter must incorporate relations to these properties. The therefrom derived material model is presented in section 4.5.

In section 2.3 on treatment options and modeling approaches relations between treatment technologies, phenomenological processes, and wastewater characteristics have been evaluated with regard to possible modeling approaches. Obviously treatment technologies for wastewater treatment are based on fundamental phenomenological processes. In contrast, most treatment technologies can not be described by modeling approaches that directly model these phenomenological processes. This is only possible for chemical processes as well as for some biological processes. In contrast, modeling of physical based treatment technologies (e.g. settling) requires the representation of space (3d) for close to phenomenological process representation. In contrast, simple model approaches (e.g. 0d) only predict treatment efficiency as function of generalized correlations.

The above problem holds also for the relation between available wastewater characterization and model approaches. Mathematical models for chemical or biological processes are based on compounds and substance and microbial properties. Conventional data on physical properties such as particle size is usually not sufficient for phenomena based modeling approaches. As consequence biological processes are represented by models as described in section 2.3.3 and separation processes are approximated by 0d model approaches as described in section 2.3.2.2. The derived process model is presented in section 4.7.

In section 2.4 knowledge representation in the context of this work refers to logic based representation languages. It has been shown that an appropriate language for knowledge representation must provide an optimal balance between expressiveness and decidability. If this balance is provided a logic based

knowledge base is able to represent complex knowledge fragments as it would not be possible by conventional data bases. As consequence for this work the representation language OWL(DL) has been selected to implement the knowledge base. OWL(DL) provides a high expressiveness and is furthermore based on standardized serialization formats such as RDF and XML.
Section section 2.5 on process modeling introduces the term model framework which is further distinguished into model concepts and model language. Model concepts provide the structural, behavioral and material view of a model. In that sense, modeling concepts define how to organize modeling objects in order to model processes and in particular chains of processes. The sequential ordering of devices and connections is described in GST. Modeling languages define the translation of modeling concepts into a formal representation. Thereby a modeling language can be based on languages for knowledge representation as described in section 2.4.
Out of the group of domain-oriented languages (section 2.5.7.3) two fundamental approaches exist which come close to meet the objective of this work. These approaches are the phenomenological modeling approach and the flowsheet based model approach. The eventually implemented approach is discussed in section 4.7.
In section section 2.6 on process design, issues related to systematic planning have been discussed. The objective of flowsheet identification has been placed into the context of problem solving. In this context a plan is a sequence of actions through problem space that connects the initial given state with possible goal states. Any algorithm for flowsheet identification must be able to efficiently search the problem space. Due to the given objective only forward search can be applied since any state can only be fully instantiated as closed chain to the initial state. Means Ends Analysis applied by Linninger and Chakraborty (1999), described in section 2.6.4, will be implemented and further developed within this work. The planning algorithm developed within this work is presented in section 4.8.

3
Material and Methods

3.1 Introduction

This chapter describes the general applied method as well as the used additional resources. The applied resources resemble to the list of used software and software libraries incorporated into the flowsheet finder developed in this work. The method reflects the steps that have been taken to achieve the objectives of this work.
The applied method is schematically shown in figure 3.1. The overall method is divided into the two main branches of model development and algorithm development. The intermediate result of model development is the knowledge base. As fundamental innovation of this work the knowledge base takes the form of an ontology serialized in OWL(DL). The process of model development is subdivided into the four steps of defining a model objective, development of conceptual models, the formalization of models and finally the serialization of the model into OWL(DL). The last two steps have been achieved by

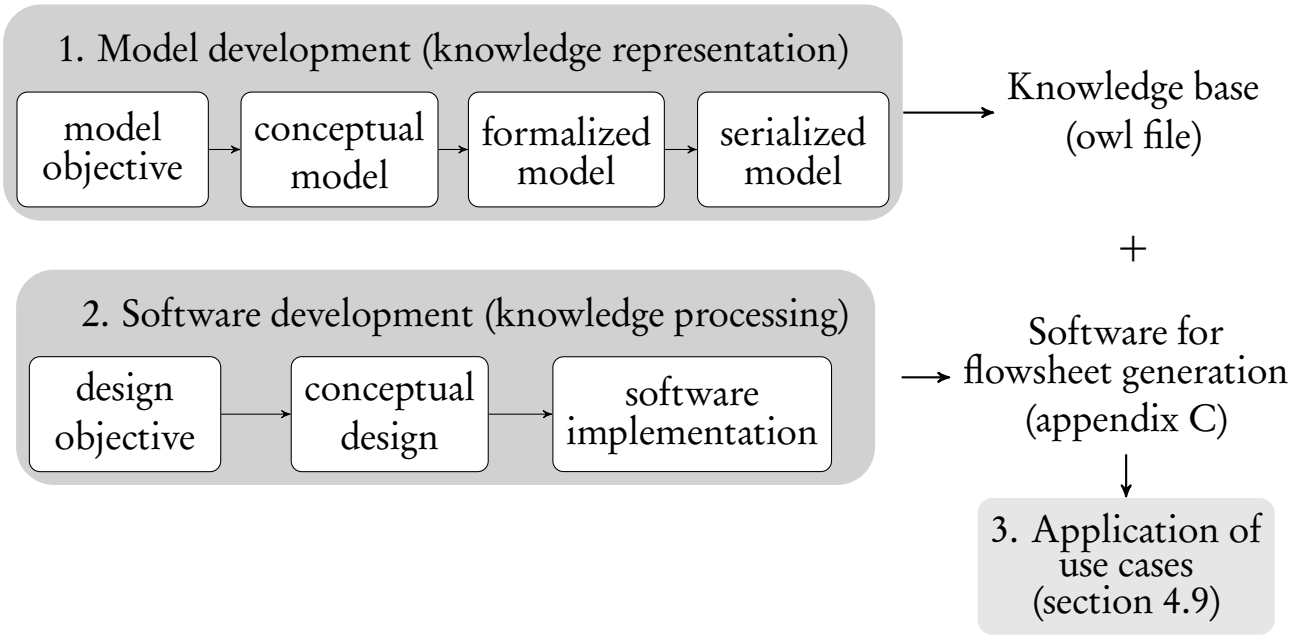


Figure 3.1: Overall applied method

using the ontology editor Protégé[1]. In particular the formalization of a model is about representing knowledge as axioms in the language of description logic. Furthermore the serialization of a model is about storing a model representation by a defined syntax, which in case of this work is set by OWL(DL). The term model in the context of this work reflects the three distinct models of material model (section 4.5), process model (section 4.7) and rules for fractionation (section 4.6). The second branch, software development, deals with the two aspects of model interpretation and the algorithm for flowsheet identification. Model interpretation is about defining how to interpret the knowledge base by the developed software. This includes for example the development of a parser to transfer the representation of a mathematical equation into an executable form. Besides model interpretation, the development of an efficient search algorithm is the major part of software development. The developed software has been implemented into the programming language Java (version 1.6) by use of the programming environment Eclipse[2]. Eventually the developed knowledge base and software have been applied on a number of use cases to show the functionality of the approach to identify reasonable flowsheets for wastewater treatment.

3.2 Used resources

The resources applied for the realization of this work can be divided into the three types of software, libraries and languages.

As mentioned in the previous section the two software applications used are the ontology editor Protégé (version 4.2) and the software development environment Eclipse (version 4.2.1).

Protégé allows for the development of ontologies based on OWL(DL). Through different panels the entities of an ontology (i.e. classes, instances, properties) can be defined (Figure 3.2). Furthermore ontologies can be visualized through different plugins such as the OWL Viz plugin (Figure 3.3).

Two model languages have been used within this work. First of all OWL(DL) has been used to codify the knowledge base developed in this work. The primary language specification of OWL is given by the W3 consortium by the reference of `http://www.w3.org/TR/owl-semantics/` and `http://www.w3.org/TR/owl2-overview/`. A profound tutorial on how to use OWL and Protégé for knowledge representation is given by Horridge *et al.* (2009). The sec-

[1] `http://protege.stanford.edu`

[2] `http://www.eclipse.org`

ond language to mention is the DOT[3] language which has been applied to visualize graphs throughout this work.

[3] http://graphviz.org

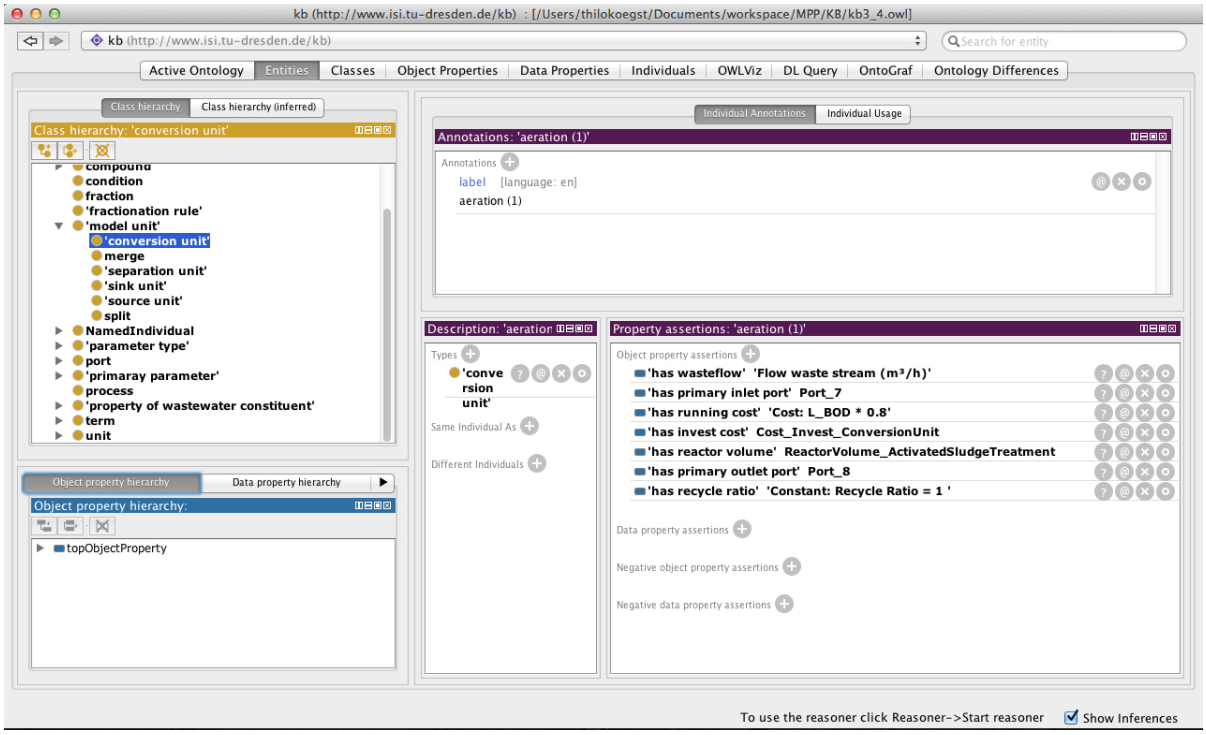


Figure 3.2: Protégé - Entities panel

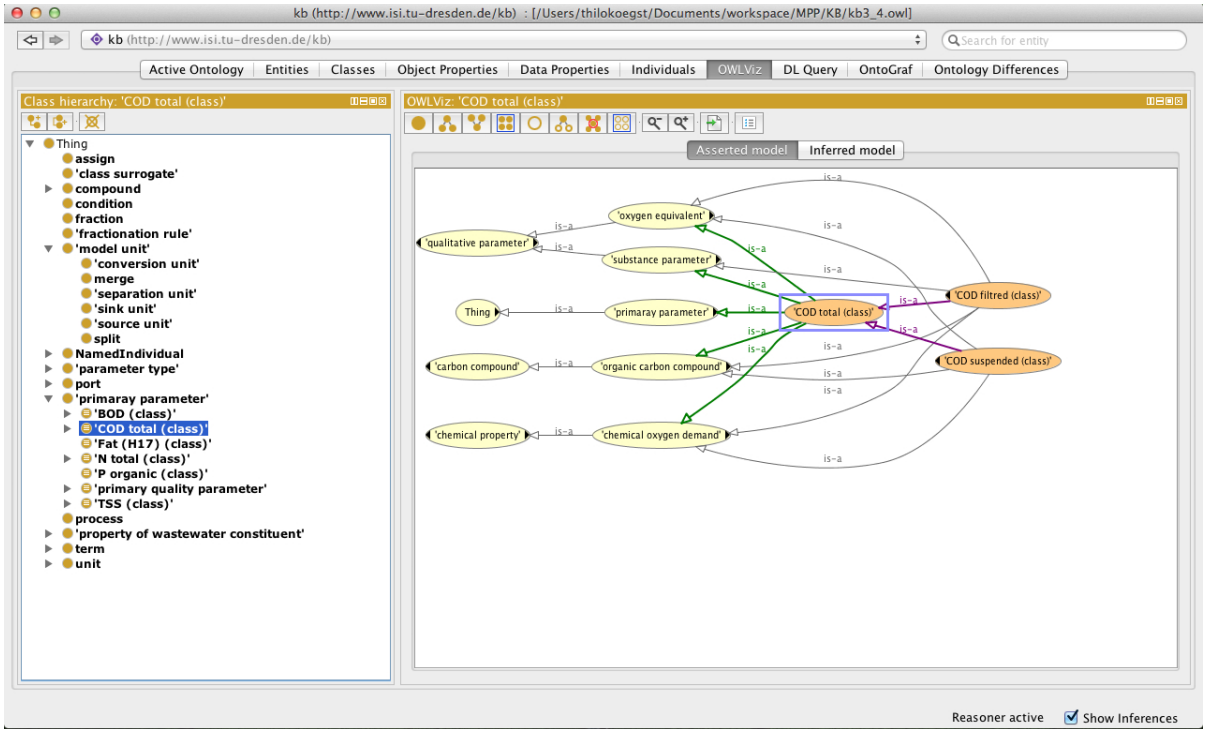


Figure 3.3: Protégé - OWL Viz panel

The developed software in the context of this work implements a number of additional libraries which have been developed and provided by other. In the following the most important of these JAVA libraries are lined out.

Jena API

A fundamental aspect of this work lies in the interpretation of models represented within an OWL coded ontology by a software to eventually identify flowsheets. The interaction between the knowledge base and the knowledge processing software is achieve by entities provided by the JENA API[4]. This API includes a query engine based on the SPARQL specification. Furthermore the API provides means for reading, processing and writing RDF data in XML format.

Hermit OWL Reasoner

The Hermit OWL reasoner[5] (i.e. HermiT) is used to determine whether or not the ontology is consistent, as well as to identify subsumption relationships between classes, and more. HermiT is the first publicly-available OWL reasoner based on a novel hypertableau calculus which provides much more efficient reasoning than any previously known algorithm.

JUNG API

The JUNG API[6] (Java Universal Network/Graph Framework) is a software library that provides a common and extendible language for the modeling, analysis, and visualization of data that can be represented as a graph or network. In particular the JUNG API has been implemented to visualize identified flowsheets in the GUI of the flowsheet finder.

ODE Solver

The ODE solver used in this work is based on an applet written by Bob Terrell[7]. The solver is based on the Runge-Kutta methods. The algorithm has been extended to be able to solve more variables as in its originate version. The ODE solver of Bob Terrell also included a string parser which has also been implemented into the flowsheet finder in order to pass equations to the ODE solver.

[4] http://jena.apache.org

[5] http://www.hermit-reasoner.com

[6] http://jung.sourceforge.net

[7] http://www.math.cornell.edu/~bterrell/de/

3.3 Definition of use cases

As the objective of this work is the development of a model and algorithm for the identification of flowsheets for given wastewater characteristics as well as given limit values for purified wastewater, the question is how to validate identified flowsheets. Usually for a developed model, the model output can be validated against measured data, in particular against measured time series. In the context of this work, this is not possible because no data exists which can be used for model validation.

A possible alternative is to compare identified flowsheets with experience from proven solutions taken from already implemented solutions. As one example, the treatment of brewery wastewater is used. Table 3.1 shows the range of wastewater characteristics of brewery wastewater. Remarkable thereby is the usually high concentration of biodegradable substances. For the purpose of direct effluent standards values as listed in table 3.2 are to be met according to German legislation.

A practical example of a flowsheet for treatment of brewery wastewater is shown in figure 3.4. This example shows the commonly applied pattern in this branch to treat high loads of particulate biodegradable matter by some anaerobic purification step (e.g. by a UASB reactor). Additionally, the flowsheet consists of a conventional Moving Bed Reactor (MBR) cycle to treat carbon as well as nitrogen compounds. For practical purposes the flowsheet furthermore contains means to remove large obstacles which could hinder the subsequent processes (i.e. fine rake) as well as an equalization tank in order to provide a near to continuous feed of the subsequent process chain. Un-

Table 3.1: Range of concentrations of brewery wastewater in mg/l

BOD_5	COD	$N_{total,anorg}$	P_{total}	settleable solids	source
1100-1500	1800-3000	30-100	10-30	10-60	Rüffer and Rosenwinkel (1985)
	1013-7775	22-1069	0.8-113	1-118	Bischofsberger *et al.* (2005)

Table 3.2: Limit values for direct discharge in mg/l (AbwV, 2004)

BOD_5	COD	NH_4-N	$N_{total,anorg}$	P_{total}
25	110	10	18	2

fortunately, wastewater characteristics and the fate of constituents after each treatment unit are not provided in literature.
In order to set up a number of use cases for testing, table 3.3 shows the definition of four sources as well as the definition of three sinks. Thereby source #1 and #2 have a moderate concentration of COD with and without additional presence of ammonia.
Source #3 and #4 have been defined by a high COD concentration again with and without contamination of Ammonia. Thereby source #4 can be interpreted as an example of a brewery wastewater.
Sink #1 reflects effluent limits for treated wastewater based on legislation as listed in table 3.2. Although the limit values for COD and N_{total} have been set to higher values due to unknown fractionation rules for brewery wastewater. Sink #2 reflects the sludge path indicated by high values for required TSS concentrations. Sink #3 is an extra sink to reflect a possible water reuse.
Use cases are defined in table 3.4. Thereby use case #4 reflects a typical wastewater of a brewery based on the values given in table 3.1.

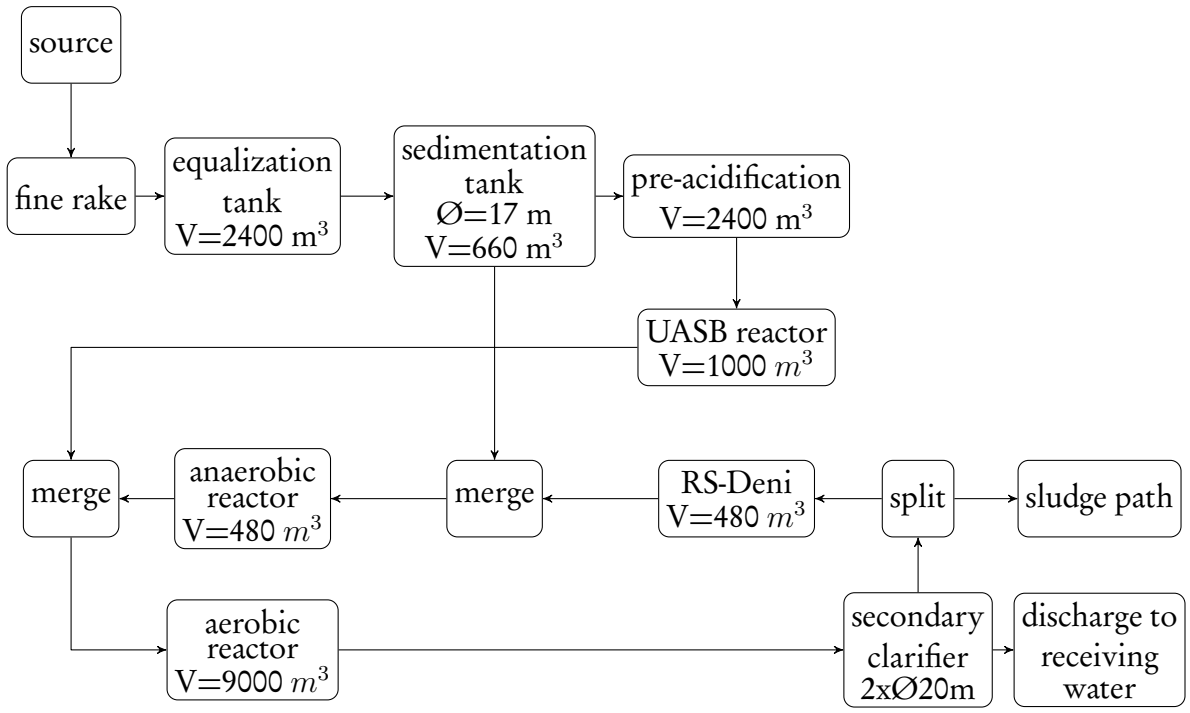


Figure 3.4: Wastewater treatment chain of a particular brewery (DWA, 2009, p. 165); hydraulic load between 2000 and 3000 m^3/d

Table 3.3: Definition of sources and sinks, concentrations in g/m^3 and flow in m^3/h

	Source				Sink		
	#1	#2	#3	#4	#1	#2	#3
Q_h	5	5	5	5			
COD	500	500	2000	1000	<500		
BOD_5			1500	500	<25	<35*	<35
NH_4		70		50	<10		
N_{total}					<40	<40*	
TSS					<10	>1000	<10

*The limit values differ from those given in table 3.2 to ease use case testing.

Table 3.4: Definition of use cases

Use case	Configuration
#1	source #1, sink #1, sink #2
#2	source #2, sink #1, sink #2
#3	source #3, sink #1, sink #2
#4	source #4, sink #1, sink #2
#5	source #2, source #3, sink #1, sink #2, sink #3

4
Results

4.1 Introduction

The practical objective of this work is the development of a planning tool to allow for conceptual design of flow sheets for industrial wastewater treatment. Following the problem analysis (section 1.3) this overall objective may be divided into the objectives to (i) develop a conceptual model framework for model representation and (ii) to develop a planning algorithm for conceptual design of treatment flow sheets. The structure of this result chapter therefore reflects this bisection. Sections 4.4 to 4.7 focus on the presentation of the developed model framework followed by the presentation of the planning algorithm starting with section 4.8. The application of the developed model and algorithm is presented in section 4.9. Section 4.2 defines a set of boundary conditions that delimit the functionality of the developed planning tool. In section 4.3 relevant terms and conventions are defined which are essential for the comprehension of the results presented.

The overall course of processed steps by the planning tool is depicted in figure 4.1. The first step requires the user specified definition of an initial plan. Each initial plan includes a list of sources and sinks (c.f. Figure 1.2). Each state outflowing of a source must be a set of primary quality and quantity parameters. Each sink may have an attributed list of preconditions to account for limit values on discharge or desired quality of reusable process water. The initial plan can be envisioned as the problem statement to be solved by the planning tool. After the initial plan has been defined no further user interaction is required in the subsequent processing. Within the second step all states of the initial plan are fractionated according to fractionation rules. Through fractionation rules wastewater characteristics of different process wastewaters may be accounted for. As result of the fractionation primary parameters are broken down into fractions of properties (e.g. amount of particulate matter).

In the last step the search algorithm explores the search space of combinations of model units from the initial plan and added model units from the knowledge

base of treatment options. A complete plan is determined by a closed graph (connecting all states to model units) and allowing for no violation of preconditions of included model units. Eventually all identified complete plans are ranked according to overall costs (i.e. the sum of investment and running cost). It follows therefrom that an optimal plan is only determined by overall minimum cost. At present model uncertainty is not addressed in any way. As a future perspective complete plans may be evaluated by a sensitivity analysis varying input parameters such as pollution load or energy cost to rank complete plans according to some measure of robustness.

An important model concept regarding flow sheet generation is the concept of a plan. Each plan can be differentiated into initial, incomplete and complete plan. A plan is represented by a graph structure of nodes and edges. Even more a plan eventually reflects a STN (Figure 4.2). Within such a STN each edge represents a wastewater stream and each node represents a task for state transition (i.e. a reactor or process). In particular a node for state transition is termed model unit within this work. Within the overall model framework states are described by the part of the material model whereas transition-nodes are described by the part of the process model.

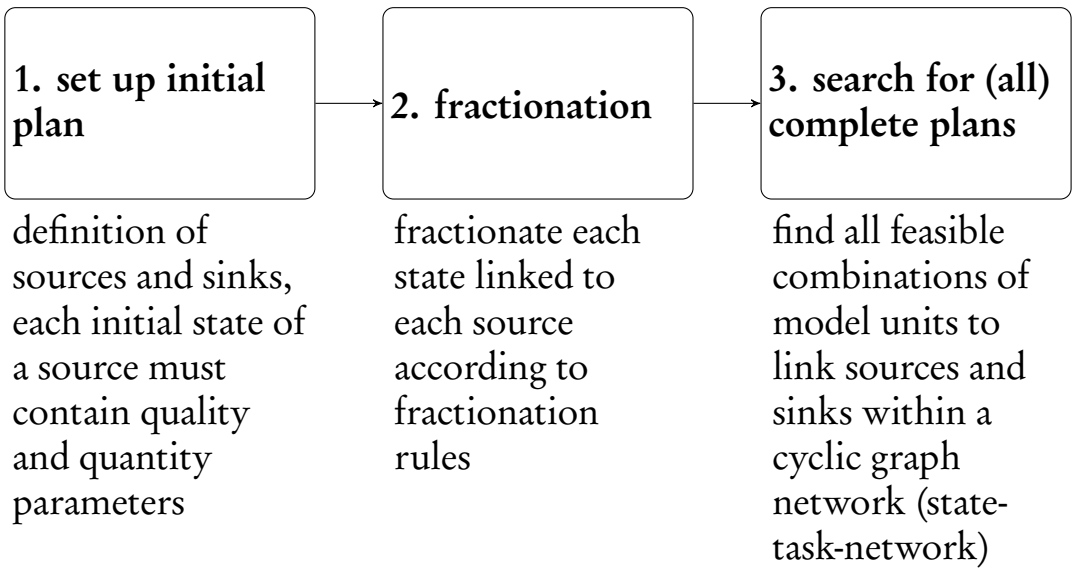


Figure 4.1: Program sequence

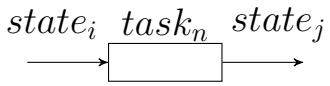


Figure 4.2: Components of a state-transition-network (STN)

A distinctive characteristic of this work is that OWL(DL) has been applied as fundamental modeling language to embrace necessary modeling knowledge. The reasons for this choice are the high degree of standardization as well as available support of auxiliary resources such as free available reasoners, editors and documentation. This circumvents the development of a handmade modeling language as described in the work of Bieszczad (2000) or the language VDDL described in Bogusch *et al.* (2001). Within the process modeling framework ONTOCape (Marquardt *et al.*, 2010) OWL(DL) has been applied to this work in the field of chemical process engineering.

The use of OWL(DL) allows for formalization of modeling knowledge starting on a conceptual knowledge level and through the use of a layered model representation also for formalization of complex model constructs. OWL(DL) provides the possibility to formalize intuitive conceptual relations nearly one to one. However the advantage of using only a single modeling language comes to the prize of a tailor made and complex model environment which is able to interpret different model constructs, see figure 4.3.

Unlike conventional model environments, models in this approach are not implemented directly but interpreted and transformed into processing algorithms at run time. Thereby the JENA API functions as interface between the OWL model and the program itself. From a database perspective the purpose

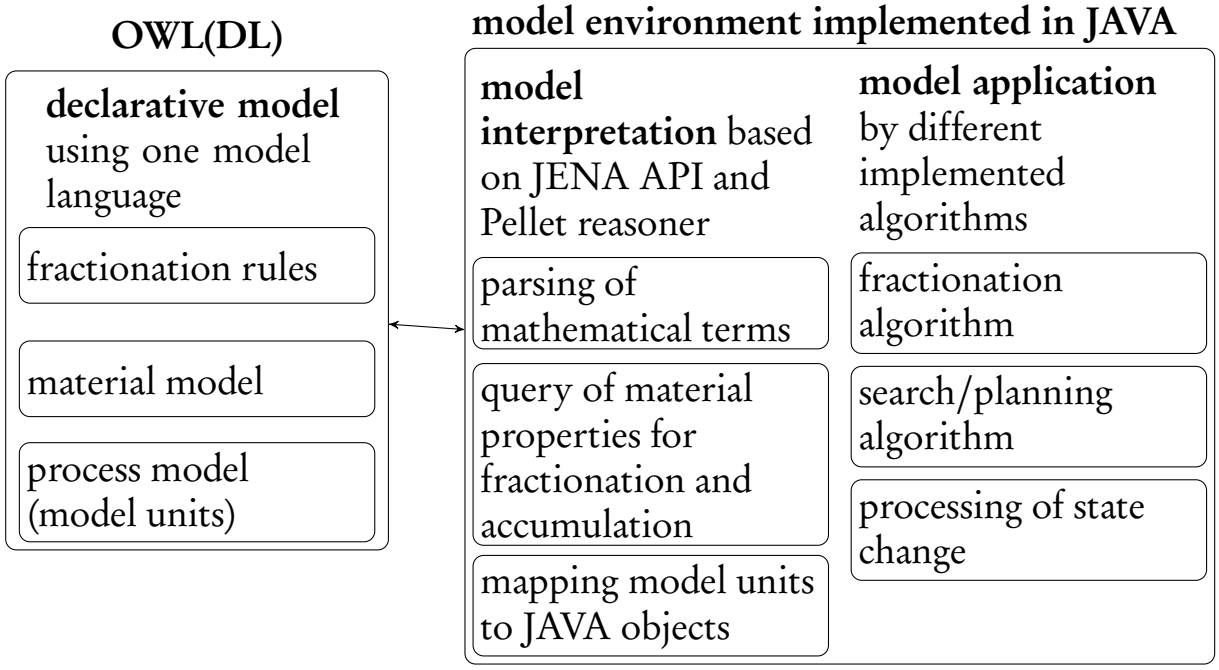


Figure 4.3: Separation of model and model application (separation of knowledge and knowledge processing)

of the JENA API is to query the OWL model to process tasks such as extracting properties of a single individual or subsuming individuals of a given concept. Beyond this algebraic notations and functions must be parsed and model units be mapped to JAVA objects[1] which can in turn be used by the search algorithm. The programming language JAVA has been used since it provides sufficient object oriented model functionality. Furthermore interfaces such as the JENA API are only available for JAVA at present.
It can be summarized that there are two innovative distinctive features introduced in the overall approach of this work. These features are:

1. the use of a single model language to provide modeling constructs on different scales of functionality (layered model representation)
2. the use of a material model to provide advanced modeling knowledge on water constituents. Through this material model

    - changes regarding a single property can be propagated to all affected fractions and
    - process models can be represented in a generic way.

Whereas the first feature has been introduced above the second is outlined as follows. Conventionally in process modeling a state is represented by a list of predefined state variables. Appendant processes, represented by mathematical equations describe the change of state variables with time or at steady state. The underlying idea of this work is to represent a state by a set of individuals. Each individual is element of an attributed set of properties represented as concepts. Eventually process description is based upon concepts that serve as placeholders comprising those individuals of a state that are element of the concept of concern.
For example a model unit representing a settling tank may predict the change of particulate matter in the over and under flow depending of the inflow. In this case particulate matter is a sole concept regardless if there are any individuals that are element of this concept or none at all. Analogue to conventional process model, concepts (precisely the subsumed individuals of concepts) serve as state variables on demand depending on the actual process.

[1]Through this programming technique data stored within the knowledge base of OWL(DL) is converted to JAVA object.

The advantage of this approach is twofold. It offers the possibility for generic model (process) description and automated model selection. Second, an individual-concept based state representation offers for a concise declaration of involved materials based on common understanding. Oppositely such a declaration is to large extent detached from requirements inflicted by process description.

4.2 Assumptions and boundary conditions

The overall objective to find feasible treatment flow sheets for industrial wastewater is simplified by the boundary condition to estimate treatment efficiency based on steady state only. Therefore input parameters of waste sources are defined by constant flow (Q_h - hourly flow rate) and constant pollutants concentrations. In practice, flow and pollution loads are subject to temporal fluctuations.
If the description of treatment processes requires the use of differential equations with respect to time the overall equilibrium state may be derived iteratively from steady state simulations. All model units within a cyclic process chain are iterated until outflowing states reach steady state.
Within the material model a wastewater stream is regarded as a two-phase system resulting from particulate and dissolved matter surrounded by a fluid phase. The discretization of properties of water constituents is fixed (e.g. fixed classes of particle sizes). The distribution regarding a single property is determined through fractionation rules which eventually reflect standardized measurement techniques. It is furthermore assumed that all fractions of a waste stream that share a single particular property equally obey to changes regarding to this property.
Within the process model units are defined on a single level of granularity, that is the level of unit operations.

4.3 Naming conventions and definitions

Following from the objective of this work the focus lies in the evolution of models through the stages of representation, interpretation and application.
An essential data structure used in this work is a plan. A plan is a set of sequentially bound states and tasks resembling a chain of treatment units from sources to discharge points. A complete plan is a treatment chain that describes a closed chain of treatment units from given waste sources to given discharge points (i.e. sinks). The desired outcome from applying the search algorithm is hence a list of complete plans. A state stands for a waste stream holding infor-

mation on quality and quantity parameters. An initial plan represents a user specified wastewater discharge by a list of sources.
A unique feature of this work is the application of an ontology for model storage. In this work an ontology always refers to a strict formalized ontology in OWL(DL). The applied model representation through OWL(DL) is based upon the three elements of class (or concept), individual and property.
In this context a concept resembles abstract or real entities which have equal properties. A concept is also referred to as class. Within an ontology concepts are organized in a taxonomy of sub- and super-classes. The original class hierarchy is termed asserted class taxonomy. A particular concept is different from a sole term since a concept can further be defined through axioms or class restrictions. In the OWL terminology a class with axioms is referred to as defined class. Individuals are members of classes. Furthermore individuals can be related through properties.
Whenever a particular concept, individual, or property is referred to in this work the element will be indicated by typewriter font. Concepts are indicated by starting with an upper case letter, e.g. `ParticulateMatter`. Individuals are indicated by starting with a lower case letter[2]. In most cases the first letter will be an `i` standing for individual or `CS` standing for class surrogate (e.g. `CSparticulateMatter`). Properties will be indicated with a lower case letter, being a predicate (e.g. `hasUnit`).
Furthermore Java objects are indicated by capitalized font with upper case first letter (e.g. QUANTITYSET) and Java methods are indicated with bold font lower case first letter (e.g. SETAMOUNT).
Commonly in process modeling a state is represented by a set of so called state variables. Each state variable is a name-value pair and the value is changed according to the formalization of processes. The set of state variables is predetermined according to the model objective. In this work a state is a set of individuals. Each individual holds a value which can represent a concentration or a flow, etc. Depending on the class membership the accumulated values of subsets of individuals amount to represented parameter and properties of this state.
Throughout this work the Manchester OWL Syntax is used for OWL notation, see table 4.1. Some important terms regarding OWL definition are listed in table 4.2.

[2]In practice the naming of individuals and concepts plays a minor role since those elements are defined through its property relations. Hence the naming conventions of this sections are only used for documentation.

4.4 Layered model representation and auxiliary concepts

4.4.1 Introduction

As introduced in section 4.1 a distinctive characteristic of the overall model approach is the development and use of a single conceptual model framework.

Table 4.1: The Manchester OWL Syntax, restrictions and boolean class constructs (Horridge *et al.*, 2006)

OWL	DL Symbol	Syntax Keyword	Example
someValuesFrom	$\exists$	some	hasChild some Man
allValuesFrom	$\forall$	only	hasSibling only Woman
hasValue	$\ni$	value	hasCountryOfOrigin value England
minCardinality	$\geq$	min	hasChild min 3
cardinality	$=$	exactly	hasChild exactly 3
maxCardinality	$\leq$	max	hasChild max 3
intersectionOf	$\sqcap$	and	Doctor and Female
unionOf	$\sqcup$	or	Man or Woman
complementOf	$\neg$	not	not Child
oneOf	x, y, z		England, Spain, Italy

Table 4.2: Terminology for OWL development and OWL design pattern (Horridge *et al.*, 2006, 2009)

primitive class	a class which does not have any class restriction $\rightarrow$ opposite: defined classes
defined class	syn.: equivalent class; a class that has at least one set of necessary and sufficient conditions
enumerated class	a class with a fixed set of individuals using the oneOf axiom
anonymous class	a class having only class restrictions but no name
universal restriction	syn.: `AllValuesFrom` restriction; restrictions that describe classes of individuals that for a given property only have relationships along this property to individuals that are members of a specified class.($\forall$)
existential restriction	a restriction that describes classes of individuals that participate in at least one relationship along a specified property to individuals that are members of a specified class.; ($\exists$)
closure axiom	also closure restriction or closed existential restriction; combine universal and existential restriction; may be written as $ONLYSOME[A, B, C]$
covering axiom	union of disjoint classes being covered
value partition	based on the covering axiom to define a discretization of properties; e.g. $Size \equiv Small \sqcap Medium \sqcap Large$
necessary condition	any class restrictions fulfilled by an individual that is not sufficient but necessary to define this class
sufficient & necessary condition	each set of necessary & sufficient conditions is an equivalent class

This approach is necessary due to the desired representation of conceptual relations between material and process model as well as fractionation rules. The representation language used as fundamental modeling language within this work is OWL(DL). From its intentional purpose OWL(DL) is designed for representation of declarative conceptual relations (i.e. representation of ontologies). This functionality is sufficient to directly represent knowledge on wastewater constituents as summarized by the material model (see section 4.5). However, complex model concepts as required for representing process models cannot directly be expressed through OWL(DL). In particular the representation of model units and fractionation rules go beyond declarative conceptual representations. Therefore required model concepts and functionality must be predefined as link for software implemented parsers and model interpreters. Eventually complex model constructs such as model units can be build from predefined fundamental model constructs, see figure 4.4. Such a resulting layered model representation is analog to the approach used in Bieszczad (2000); Bogusch *et al.* (2001); Marquardt *et al.* (2010).
Essential fundamental model constructs are the `Term` and `Assign` concept to represent mathematical notations and functions, the `Condition` concept to represent conditional relations and the `Process` concept to represent conversion and separation processes. Within this section the purpose and construction of these fundamental model constructs is presented in detail.

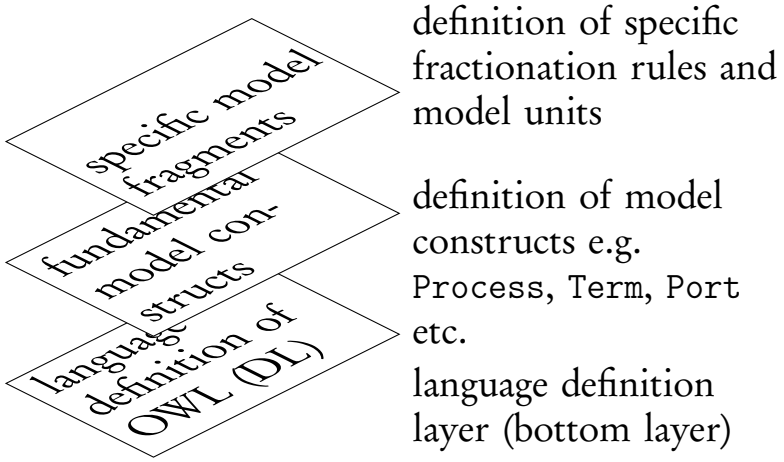


Figure 4.4: Layered model representation

4.4.2 Representing algebraic terms

A fundamental prerequisite for the formalization of process models in this work is the ability to represent mathematical equations. This necessity is not provided by OWL(DL) and it's underlying concepts from description logics.

This deficit has been circumvented by defined concepts for sole representation of mathematical notations whereas the interpretation and solving is left to a Java implemented recursive parsing algorithm. In particular the concept of `Term` and `Assign` are used to represent mathematical terms and variable value allocations. A single term thereby represents a single mathematical operation and two involved operands. As consequence, an arbitrary equation may be represented by a set of terms resulting in a binary tree. A binary term has two associated properties that are `hasLeftSide` and `hasRightSide` (e.g. sum, product etc.). A unary term has only the property `hasRightSide` attached (e.g. logarithms). A constant holds a value only. Most important, variables represent either a particular variable or a class surrogate (see section 4.4.4).

A term is formally represented by a single individual of concept `Term`. The `Assign` concept is required for variable value allocations (i.e. $y = f(...)$). The class restrictions for the `Term` and `Assign` concept are depicted in propositions 4.1 to 4.5.

$$\texttt{Term} \equiv \texttt{BinaryTerm} \sqcup \texttt{UnaryTerm} \sqcup \texttt{Constant} \tag{4.1}$$

$$\texttt{BinaryTerm} \equiv (\texttt{Divide} \sqcup \texttt{Multiply} \sqcup \texttt{Minus} \sqcup \texttt{Plus} \sqcup \texttt{Power} \sqcup \texttt{Exp}) \sqcap \exists \texttt{hasLeftSide.Term} \sqcap \exists \texttt{hasRightSide.Term} \sqcap \texttt{hasSide} = 2 \tag{4.2}$$

$$\texttt{UnaryTerm} \equiv \texttt{Log} \sqcap \exists \texttt{hasRightSide.Term} \sqcap \texttt{hasSide} = 1 \tag{4.3}$$

$$\texttt{Constant} \equiv \exists \texttt{factor}.\mathit{double} \tag{4.4}$$

$$\texttt{Assign} \equiv \exists \texttt{hasResult}.(\texttt{Term} \sqcup \texttt{Parameter}) \sqcap \exists \texttt{hasRightSide}.(\texttt{Term} \sqcup \texttt{Parameter}) \tag{4.5}$$

To ensure clearness a closure axiom is added to the above propositions. The exact domain and range specifications of the introduced properties are listed in table 4.5. The use of the `Term` and `Assign` concepts is exemplary shown in figure 4.5. Thereby the equation $z = 2x + y^2$ is represented by three terms. The term t_1 is the root of the binary tree. The value assignment to variable z is achieved by the individual a_1 of concept `Assign`.

A similar form of representing mathematical equations has been presented and used by Bogusch and Marquardt (1997); Morbach *et al.* (2007a, 2008c).

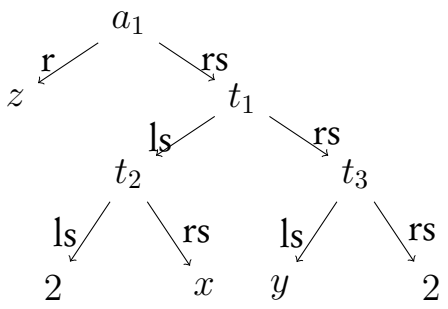


Figure 4.5: Binary tree representing equation $z = 2x + y^2$ with $a_1 \in$ `Assign`, $t_1 \in$ `Plus`, $t_2 \in$ `Multiply`, $t_3 \in$ `Power`, $\{x, y, z\} \in$ `Variable`; the symbols r, ls, and rs represent the properties `hasResult`, `hasLeftSide`, and `hasRightSide`

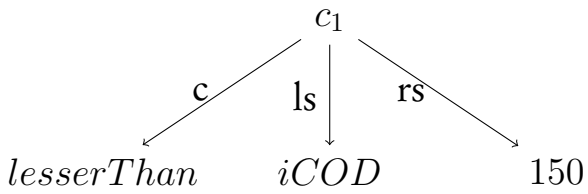


Figure 4.6: Representing the condition $COD < 150$ mg/l with $c_1 \in$ `Condition`, $lesserThan \in$ `Comparator`, $150 \in$ `Constant`; the symbols c, ls, and rs represent the properties `hasComparator`, `hasLeftSide`, and `hasRightSide`

4.4.3 Representing conditional terms

Similar to the problem to represent mathematical equations also the representation of logical conditions is not designated by OWL(DL). Therefore may conditions of type "**if** parameter < value **than** do ..." be expressed through the `Condition` concept. Analogous to the `Term` concept, the `Condition` concept connects two operands via the properties `hasLeftSide` and `hasRightSide`. The relational operator is attached via the property `hasComparator`. Thereby, the relational operators are subsumed by the enumerated class `Comparator`. The respective class restrictions are shown by propositions 4.6 and 4.7.

$$\texttt{Comparator} \equiv \{\texttt{equal}, \texttt{largerThan}, \texttt{lesserThan}\} \tag{4.6}$$

$$\begin{aligned} \texttt{Condition} \equiv\ & \exists \texttt{hasComparator.Comparator} \sqcap \\ & \exists \texttt{hasLeftSide.Term} \sqcap \exists \texttt{hasRightSide.Term} \sqcap \\ & \texttt{hasSide} = 2 \end{aligned} \tag{4.7}$$

The formal representation of an exemplary condition is depicted in figure 4.6. Units are preserved through the membership of quality parameter to concept `SubstanceParameter`. From this concept all sub-concepts inherit the relation to the unit g/m^3 via the property `unit`.

4.4.4 Class surrogates

A fundamental aspect offered through the use of ontologies is the declaration of concepts by generalization and specialization. Thereby any concept may be further described by one or more detailed concepts or subsumed by a more general one. In OWL(DL) this is supported through the definition of class taxonomies. A desired feature for the material model is the ability to address concepts rather than particular individual entities for model representation. For example, the precondition of a sink may define a maximum concentration of COD (addressing the individual parameter COD). In contrast may the regarded wastewater state hold a number of fractions that contribute to the total concentration of COD (addressing the concept COD). Through the material model it must be possible to subsume the total concentration of the parameter COD from all related fractions.

In OWL(DL) instances of classes can be related through properties. In contrast classes can have no properties attached[3]. A desired feature of the model representation is hence not supported in OWL(DL). For example a model unit may have a precondition to limit the amount of particulate matter of the in flowing stream. Thereby can the condition be formulated as explained in the above section. However particulate matter as a substance property is expressed as a concept (see section 4.5.2.2) and not by an individual.

To overcome this shortcoming concepts may have an additional surrogate attributed. Such a surrogate is an individual which holds an additional membership to the class `ClassSurrogate`. Through this definition there follow two distinct cases.

As simple case an individual may solely represent a class, reflected through its membership to `ClassSurrogate`. This individual cannot have an attributed value but serves as nominee for the surrogated class only. In figure 4.7 the individual `csParticulateMatter` depicts such a case. If the individual `csParticulateMatter` is addressed within the model (e.g. within a condition or term, etc.) all individuals which hold membership to the concept `ParticulateMatter` are subsumed to induce their cumulated value (e.g. the value attached to `iVSS`). This value can then be processed (e.g. within a condition, term, etc.).

[3]In short a class may have class restrictions (axioms) but cannot be linked to another class or instance via a property. Only individuals can be linked as a triple by a property relation; see work group note `http://www.w3.org/TR/swbp-classes-as-values/`

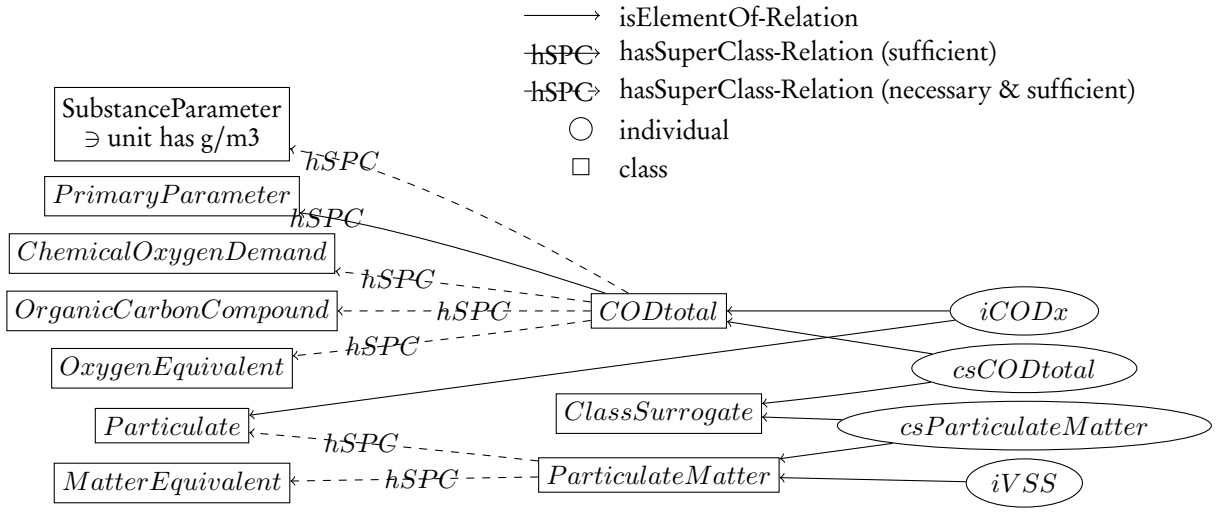


Figure 4.7: Exemplary use of `ClassSurrogate` concept

In a second case an individual may not only be part of the `ClassSurrogate` concept but also be defined as primary parameter (`PrimaryParameter` concept). In this case the individual may have an attributed value by itself or serve as nominee for the surrogated class. In figure 4.7 the individual `iCOD` reflects such a case. For example may the class surrogate of COD hold the value of the total concentration of COD in the initial plan. After fractionation this value is fully assigned to fractions of COD.

4.4.5 Units

Units are accounted for in the declarative model by standard units. Thereby quality parameters are defined as concentration and hydraulic parameters are described as flow rate. Internal, units are only accounted for by a standard dimension which is g/m^3 for concentration and m^3/h for flow rate. In the case that parameter in the initial plan are specified by dimensions other than standard dimensions they are converted (see table 4.3). To automatically assign a unit to a quality parameter the concept of `SubstanceParameter` has the class restriction $\ni$`hasUnit` g/m^3, respectively hydraulic parameter representing a flow rate have the class restriction $\ni$`hasUnit` m^3/h.
Since the representation of algebraic terms is not based on the concept of scalar variables (as described in Bogusch and Marquardt (1997)) the correctness of a term must be guaranteed while it is implemented into the model.

4.4.6 Representing processes

In contrast to phenomenological modeling approaches (c.f. section 2.5.7.3 on page 79) processes related to state change are attached to model units (i.e. the reactor itself) and not to substances and compounds. For the description of

Table 4.3: Units accounted in the declarative model

Type of unit (Sub concept of `Unit`)	**Dimension** (instance)	**Factor** (data property of type float)
`Concentration`	g/l	1000
	g/m^3	1
	mg/l	1
`Flowrate`	l/h	1000
	m^3/h	1
`Dimensionless`	-	-

processes related to treatment of wastewater this is the only feasible approach since no generic function exists to express a single process in terms of educts and contextual boundary conditions. This in turn is related to the fact that educts and products of reactions of wastewater treatment processes are mostly not identifiably in terms of specific chemical substances and compounds.

According to the definition of model units (section 4.7.2) there are two basic types of processes, conversion and separation processes. Beyond these two also merging of two states and the splitting of a single state into two can be understood as processes. However merging and splitting are only defined internal by alligation and hence are not represented in the declarative model.

All separation processes are defined in the form of a simple point settler (0D). Thereby the separation efficiency regarding a particular compound X_{feed} in the feed is defined by a single parameter f_e. Hence the concentration of the underflow and effluent, $X_{underflow}$ and $X_{effluent}$ is described as $Q_{feed}X_{feed}f_e = Q_{effluent}X_{effluent}$ and $Q_{feed}X_{feed}(1 - f_e) = Q_{underflow}X_{underflow}$. After rearranging, the concentration of effluent and underflow is derived by the following equations:

$$X_{effluent} = f_e \cdot X_{feed} \cdot \frac{Q_{feed}}{Q_{effluent}} \tag{4.8}$$

$$X_{underflow} = \frac{X_{feed} \cdot Q_{feed} - X_{effluent} \cdot Q_{effluent}}{Q_{underflow}} \tag{4.9}$$

The use of a simple point settler for all separation processes is surely a huge simplification. Nevertheless for the phase of preliminary design of flow sheets this choice seems reasonable.

For the consideration of conversion processes the general conversion model r_F has been applied. Taking into account a stoichiometric coefficient v_F and a reaction rate ρ (equation 4.10). Time dependency is reflected by the mass

Table 4.4: Heterotrophic Growth according to Olsson and Newell (1999, p. 58)

Stoichiometric Terms activated sludge (AS)	substrate (S_S)	**Rate***
1	$\frac{-1}{Y_H}$	$\hat{\mu} \cdot AS \cdot \frac{S_O}{K_{OH}+S_O} \cdot \frac{S_S}{S_S+K_S}$

*S_S and S_O relate to the entities of $csBIO$ and $iOxygen$ in figure 4.8

Listing 4.1: Result of the recursive parsing of process growth of heterotrophic biomass

```
1 Assign: X_Het_OE == 1
2 Assign: BODu_S_Fraction == (-1/0,67)
3 Rate: (6 (X_Het_OE ((5/(0,2+5)) (BODu_S_Fraction/(35+BODu_S_Fraction)))))
```

balance for an ideally mixed reactor (CSTR) embedded in equation 4.11. Although the overall approach is based on continuous flows the build up of organisms related to treatment of organic pollution may only be represented by differential equations. Using the Runge-Kutta method differential equations such as equation 4.11 are iteratively solved until equilibrium state is reached.

$$r_F = v_F \cdot \rho \tag{4.10}$$

$$\frac{d}{dt}(VC) = Q_{in}C_{in} - Q_{out}C_{out} + r_F V \tag{4.11}$$

Within the declarative model, processes are represented by the `Process` concept. This concept is defined on the concepts of `Assign` and `Term` as well as the properties `hasStoichometricTerm` and `hasRate`.

$$\texttt{Process} \equiv \exists \texttt{hasRate.Assign} \sqcap \exists \texttt{hasStoichiometricTerm.Term} \tag{4.12}$$

The exemplary representation of the conversion process given in table 4.4 is shown in figure 4.8. The individual `iHeterotrophicGrowh` being a member of concept `Process` has two property relations via the property `hasStoichemetricTerm` and one relation via the property `hasRate`. The algebraic terms as defined in table 4.4 specifying the stoichiometry as well as the reaction kinetics are translated according to the `Term` concept as binary tree.
The result of the recursive parser that interprets the binary tree representation of equations is shown in listing 4.1 for the process shown in figure 4.8.

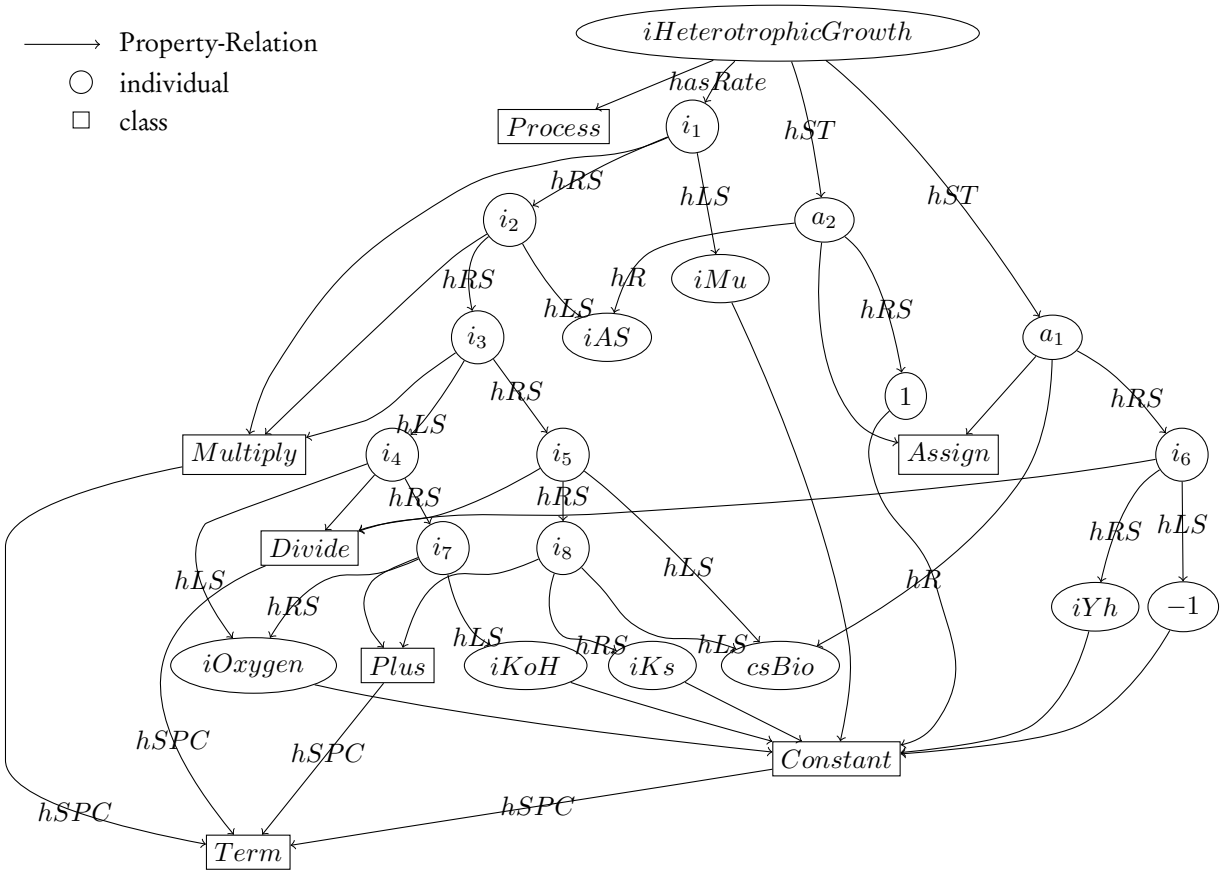


Figure 4.8: Graph of OWL representation of heterotrophic growth (see table 4.4), with $a_i \in Assign$ and $i_j \in Term$ using the following property-relations: hSt-hasStoichiometric Term, hasRate-hasRate, hR-hasResult, hSPC-hasSuperClass, hLS-hasLeftSide, hRS-hasRightSide

The applied model functionality regarding the representation of processes accepts two fundamental disadvantages.

First, state change can only take place as defined in a particular model unit through its attached process descriptions. That means that an occurring process is not the result of the presence of educts within a state but only by the definition of processes itself. In the example above (Figure 4.8) the process of growth of heterotrophic biomass may only take place providing sufficient concentration of dissolved oxygen and the absence of any inhibiting substances. Whereas the presence of oxygen may be a predefined assumption of an aeration reactor the second, the absence of any inhibiting substances can only be taking into account by a strict definition of preconditions of the regarded model unit. A second example may be the merging of two waste streams resulting of a drastic change on pH value which in turn may cause the change of equilibrium between substances or the precipitation of a substance from dissolved to particulate occurrence of a substance. Reactions of this kind are not accounted for

Table 4.5: properties defined within auxiliary concepts

Property	Domain	Range
hasLeftSide	Condition ⊔ Divide ⊔ Minus ⊔ Multiply ⊔ Power	Term ⊔ SubstanceProperty ⊔ QualityParameter ⊔ QuantityParameter
hasResult	Assign	GeneralQualityParameter ⊔ Fraction⊔ Ratio⊔ SolidsParameter ⊔SubstanceParameter⊔ Property⊔ Term
hasRightSide	Condition ⊔ Divide ⊔ Minus ⊔ Multiply ⊔ Power	Term ⊔ SubstanceProperty ⊔ QualityParameter ⊔ QuantityParameter
hasComparator	Condition	Comparator
hasUnit	ParameterType	Unit

automatically by the material model and hence must be intercepted though process definitions of model units.
The second disadvantage results from the incorporation of biological reactions resulting in the use of wastewater specific constants such as growth rate or half-saturation constant. A general use of constants of this type is questionable since they must be derived from particular analysis of regarded wastewaters. In the scope of this work, parameter of this kind have been derived from literature sources. For future applications values of these reaction parameters can be adjusted as qualitative functions of types of industrial branches.

4.4.7 Properties

In the latter sections fundamental concepts have been defined which are used to eventually form desired model constructs. In table 4.5 all properties which are used for the definition of these concepts are listed. The domain and range specification defines between which concepts the use of a particular property is allowed.

4.5 Material model

4.5.1 Introduction

The necessity of a material model[4] within this work emerges from the deviation between what is expressed by quality parameters for wastewater characterization and what is required for phenomena based process description. In contrast to material models or the particular view on material information of

[4]The term material model in this context has been introduced by Linninger *et al.* (1996b). In the work of Linninger *et al.* (1996b) a Material Model emulates context-specific description of material mixtures and their thermodynamic properties through a formal representation.

a model as introduced in section 2.5.6.4 and the material concept within this work lies in the attempts to bridge the gap between what is represented by measurable parameter of water quality and what substances and compound mixtures are present within a waste stream.

Commonly quality parameters are based upon easy deducible biological or physico-chemical properties of wastewater constituents. It follows therefrom that assertion on different parameters overlap. The major requirement on the model concept is therefore the ability to account for interdependencies between quality parameters.

The design objective is to represent a state of wastewater stream in terms of independent fractions of matter. The independence between fractions is defined by a unique allocation of properties for each fraction. The origin of properties is defined by the three perspectives of quality parameter, compounds & substances, and substance properties. A single quality parameter represents either a particular substance (e.g. Nitrate), a compound group (e.g. Total Nitrogen) or a particular property (e.g. Total Suspended Solids), see figure 4.9. Independent from parameters, compounds and substances may be related to substance properties (e.g. organic carbon compounds are chemical oxidizable; the substance ethanol occurs in liquid phase at room temperature). The relation between compounds or substances to properties are context dependent (e.g. dependent on temperature, pH value etc.)

In this work fractions of matter are represented by individuals in OWL(DL). It follows that a state of a wastewater stream is describe by a set of individuals. These individuals populate classes that represent properties of these individuals. This approach is similar to an object oriented programming paradigm since properties are also inherited through class taxonomy.

Oppositely to the representation of model units (see section 4.7 on page 141) the definition of material concepts is based on class taxonomy which is supported by the expressiveness of OWL(DL). Simplified, knowledge on materials can be mapped one to one into OWL(DL).

4.5.2 Concept definition by class taxonomy

As described in the introduction, there are three root categories for material concepts: parameter, compounds & substances, and properties. Each of these categories is further described in this section.

A new parameter can only be added to the model if it is different to the ones already included. In other words every parameter is unique with regard to its

asserted properties. In turn two parameters being member of an identical set of properties are equal.

4.5.2.1 Defining parameter types, compounds and substances

Parameters for wastewater characterization are organized according to general parameter types. The skeletal structure for defining parameter are subconcepts below the concept `ParameterType`, see table 4.6. Concepts subsumed by concept `ParameterType` have no class restrictions but are delimited by the definition of disjoint classes. For example any parameter of type solids parameter (concept `SolidsParameter`) cannot be of type substance parameter (concept `SubstanceParameter`) as expressed by
$\texttt{SolidsParameter} \equiv \neg\texttt{SubstanceParameter} \sqcap \bot$.
Any parameter of concept `QualityParameter` which is not a `GeneralQuality Parameter` may further be described by the concepts of `Compound` and `Substance`. Thereby the concept `Substance` enfolds all parameter that represent a direct substance with known molecular composition and substance properties. In turn the concept `Compound` groups substances and compounds which belong to equal chemical subtypes (e.g. nitrogen compounds, carbon compounds etc.).

4.5.2.2 Defining substance properties

The exploitation of substance properties of wastewater constituents is essential for treatment of wastewater. Therefore the ability to consider substance properties within a material model is essential. Through the material model generic knowledge on the relation of biological, physical and chemical properties related to water quality has been captured. Related to physical properties referencing of particle density and particle size is most important. With respect to biological properties the biodegradability in combination with favorable boundary conditions (presence/absence of dissolved oxygen, temperature

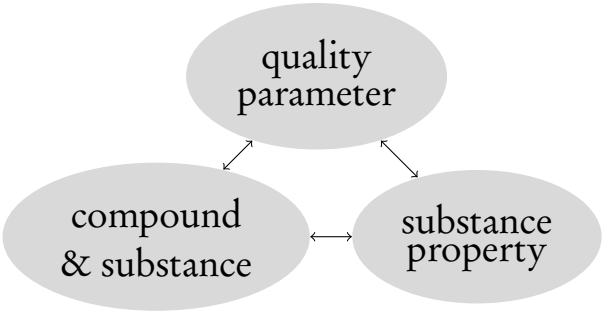


Figure 4.9: Three categories for material description

Table 4.6: Sub-concept under the concept ParameterType

Concept	Complement	Description
Quality Parameter	¬ Quantity Parameter	covers all parameter for qualitative characterization of wastewater
Quantity Parameter	¬ Quality Parameter	covers all parameter for quantitative characterization of wastewater (e.g. flow rate); Loads can not primarily be defined but can be expressed as product of concentration and flow
General Quality Parameter		covers quality parameter which are not substance specific but enfold context specific information (e.g. temperature, pH value)
Substance Parameter	¬ Solids Parameter	in contrast to General Quality Parameter this concept enfolds parameter that represent substance or compound specific constituents (e.g. Total Phosphorous, Nitrate, Biological Oxygen Demand)
Solids Parameter	¬ Substance Parameter	covers parameter that characterize a wastewater state by the sum of solids (e.g. Total Suspended Solids)
Matter Equivalent	¬ Oxygen Equivalent ⊓ ¬ Carbon Equivalent	covers parameter where a related mass or concentration maps direct the substance or compound of concern (e.g. Nitrate)
Oxygen Equivalent	¬ Matter Equivalent ⊓ ¬ Carbon Equivalent	covers parameter where a related mass or concentration maps indirectly to the substance or compound of concern by another substance, in this case oxygen (e.g. Chemical Oxygen Demand)
Carbon Equivalent	¬ Matter Equivalent ⊓ ¬ Oxygen Equivalent	covers parameter where a related mass or concentration maps indirectly to the substance or compound of concern by another substance, in this case carbon (e.g. Total Organic Carbon)
Parameter Ratio		covers parameter ratios
Primary Parameter		a helper concept that solely subsumes all quality parameter (i.e. BOD, COD, TSS, etc.)

etc.) must be considered. Chemical parameters are usually substance or compound specific. Important chemical reactions are precipitation with respect to solubility or induced by pH value.
Within this work a differentiation has been made between contextual properties and substance properties. The first refers to properties that are imperative for an overall state such as temperature, pressure, pH value, alkalinity. Although parameters such as pH value or alkalinity originate from the presence of constituents within the waste stream they are not accounted as substance properties. Within this work the dependence on contextual properties has been disregarded by assuming a constant temperature range as well as by

a constant normal pressure. This is justified by the occurrence of common treatment objectives under these boundary conditions. In contrast substance properties reflect properties of isolatable fractions of substances such as being either dissolved or particulate. Substance properties refer to properties that can be allocated to particular compounds, substances or fractions.
Within the material model of this work substance properties have been organized in a class taxonomy as depicted in figure 4.10. This taxonomy covers at present only basic parameters and is not exclusive. Analogue to the already captured parameters further parameters can be added.

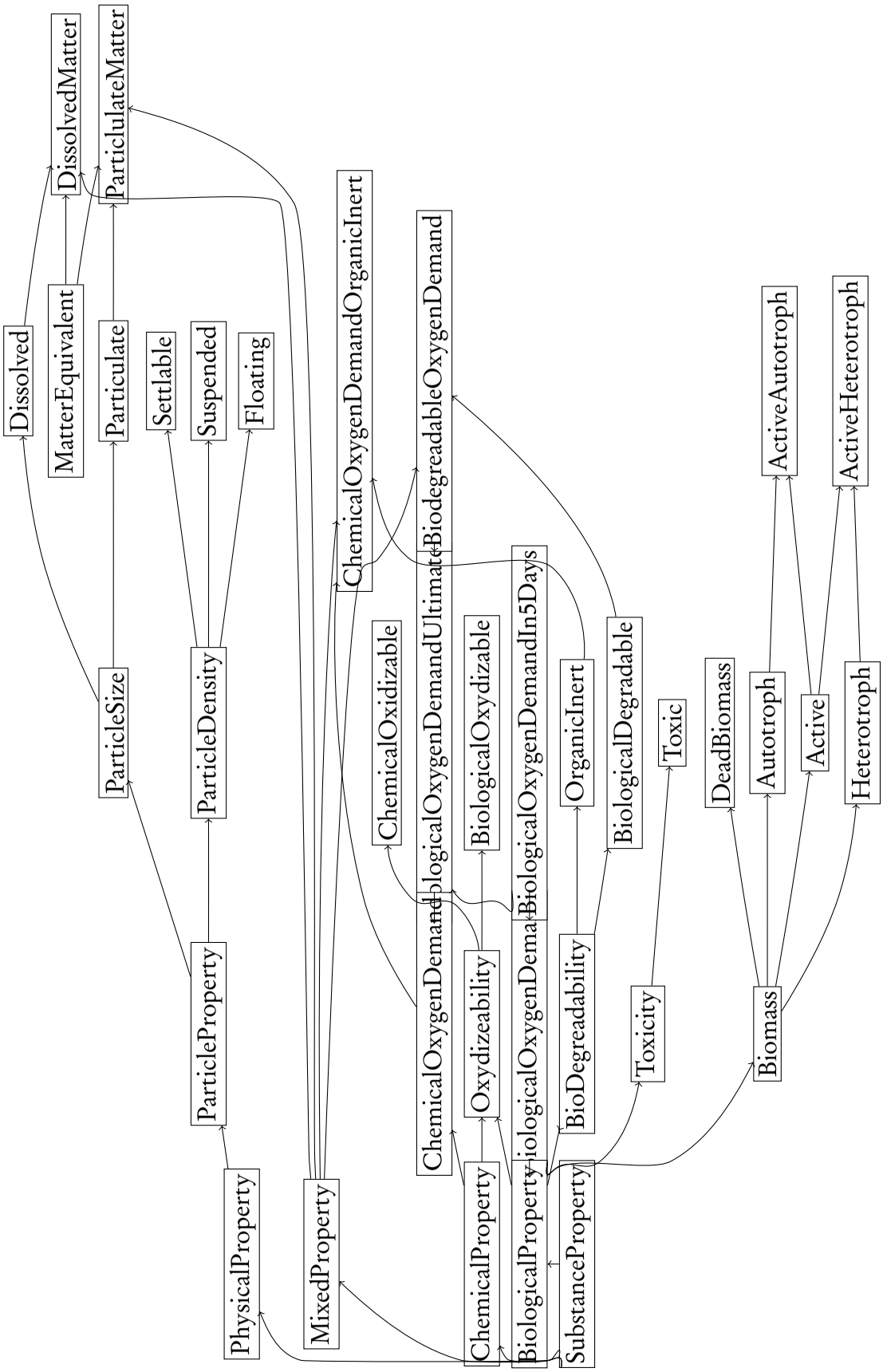


Figure 4.10: Asserted class taxonomy of concept `SubstanceProperty`

Whereas the concepts of parameter, compound, and substance are of purely categorical nature, substance properties are in general continuous. In concession to the overall approach and the usage of OWL as modeling language ranges of substance properties cannot be mapped as continuous scales. The resulting discrete representation is eventually based on standardized methods from wastewater analytics (see section 2.2).

As for most modeling objectives regarding wastewater treatment there is a problem to essentially map particle size distribution due to high analysis effort. Hence assertion on particle size is very course allowing commonly only a distinction between dissolved and particulate substances. For example a measurement on TSS refers to all constituents that are above a defined particle size, dependent on the applied filter. This means that the continuous range of particle size is divided according to an interval scale with two ranges. The interval is determined by pore size of the chosen filter material.

Based on expert knowledge from literature industry specific wastewater characterizations may be fractionated into more fine grained property structures defining industry specific fractionation rules. Hence it may be possible to account for colloids within the material model.

Similar to parameter types final property categories are delimited from adjoining ones by disjointness. For example, the concept of `Particulate` is disjoint from `Dissolved` and vice versa. This underlines the fact that a particulate substance cannot be dissolved at the same time.

4.5.2.3 Defining quality parameter

Quality parameters are defined as intersection between sub-concepts of `ParameterType`, `Substance`, `Compound`, and `SubstanceProperty`. In the definition of quality parameter it is essential to use necessary & sufficient conditions (i.e. defined class or equivalent class). Only then it is assured that an individual that holds membership to the required property concepts is equivalent to the desired quality parameter.

By the use of intercepting concepts various quality parameter have been defined. The inferred class taxonomy regarding the material model is shown in figure 4.11. New properties with arbitrary value partitions can be defined. The types of properties and value partitions depends (i) on the ability to formulate models that incorporate these properties; and (ii) on the ability to provide information to describe quality parameter according to the defined class taxonomy.

Example 1 - Definition of COD and TKN
For example, the parameter COD_{total} is defined by the concept `COD_total` as shown by proposition 4.13. The COD measurement of a filtered probe $COD_{filtered}$ is represented by the concept `COD_filtered` as formulated in proposition 4.14. The only difference between the concepts of `COD_total` and `COD_filtered` is that the latter one additionally intercepts the concepts `Dissolved`. The formalization of proposition 4.13 in OWL using RDF/XML syntax is shown in listing 4.2.
As another example proposition 4.15 shows the definition of the quality parameter TKN. This proposition literately states, that any individual that is either member of concept `Ammonia` or `Norganic` and additionally holds membership to `MatterEquivalent` and `SubstanceParameter` is equivalent to the concept `TKN`. The de factor formalization of proposition 4.15 in OWL using RDF/XML syntax is shown listing 4.3.

Analogue to example 1 additional basic quality parameters have been captured in the material model as shown in figure 4.11. Through the reasoner concepts are reorganized according to their class restrictions from an asserted class taxonomy into an inferred class taxonomy.

$$\begin{aligned} \texttt{COD\_total} \equiv\ & \texttt{ChemicalOxidizable} \sqcap \\ & \sqcap \texttt{OxygenEquivalent} \sqcap \texttt{QualityParameter} \\ & \sqcap \texttt{SubstanceParameter} \sqcap \texttt{OrganicCarbonCompound} \end{aligned} \tag{4.13}$$

$$\begin{aligned} \texttt{COD\_filtered} \equiv\ & \texttt{ChemicalOxidizable} \\ & \sqcap \texttt{QualityParameter} \sqcap \texttt{SubstanceParameter} \\ & \sqcap \texttt{OxygenEquivalent} \sqcap \texttt{Dissolved} \\ & \sqcap \texttt{OrganicCarbonCompound} \end{aligned} \tag{4.14}$$

$$\begin{aligned} \texttt{TKN} \equiv\ & \texttt{MatterEquivalent} \sqcap \texttt{SubstanceParameter} \\ & \sqcap (\texttt{Ammonia} \sqcup\ \texttt{N\_organic}) \end{aligned} \tag{4.15}$$

Listing 4.2: RDF/XML rendering of concept COD in the OWL file (based on proposition 4.13)

```
<owl:Class rdf:about="#COD">
  <rdfs:comment rdf:datatype="&xsd;string"></rdfs:comment>
  <rdfs:label rdf:datatype="&xsd;string">COD total (class)</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PrimaryParameter"/>
  </rdfs:subClassOf>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#ChemicalOxygenDemand"/>
    <owl:Class rdf:about="#OrganicCarbonCompound"/>
    <owl:Class rdf:about="#OxygenEquivalent"/>
    <owl:Class rdf:about="#SubstanceParameter"/>
  </owl:intersectionOf>
</owl:Class>
</rdf:RDF>
```

Listing 4.3: RDF/XML rendering of concept TKN in the OWL file (based on proposition 4.15)

```
<owl:Class rdf:about="#TKN">
    <rdfs:label xml:lang="en">TKN (class)</rdfs:label>
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#MatterEquivalent"/>
                <rdf:Description rdf:about="#SubstanceParameter"/>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="#Ammonia"/>
                        <rdf:Description rdf:about="#N_organic"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#N_total"/>
</owl:Class>
```

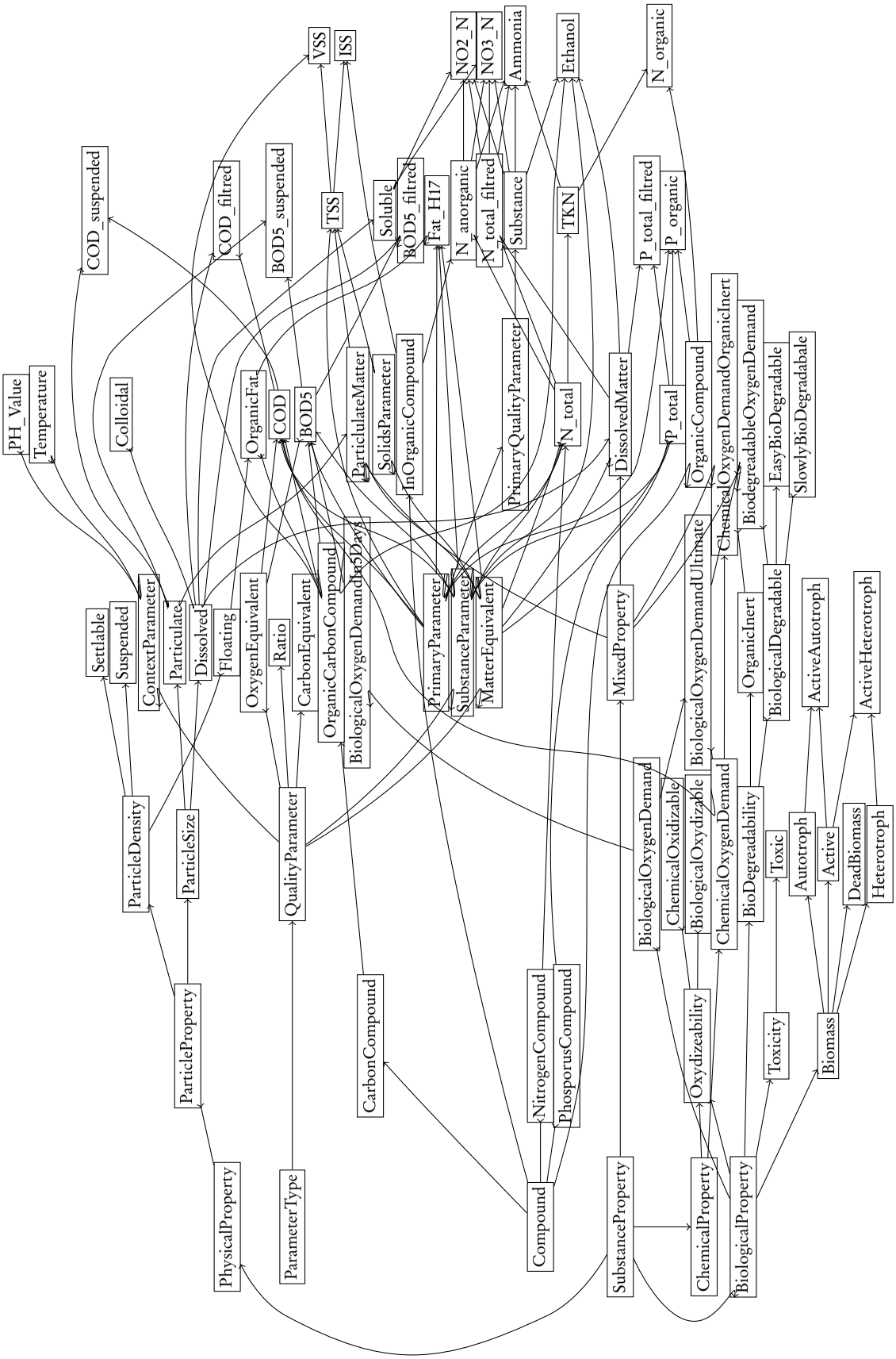


Figure 4.11: Quality Parameter - Inferred class taxonomy

4.5.3 Individuals and properties

The introduced class taxonomy in the section before serves as backbone of the material model. Applying this taxonomy as domain ontology allows already the extraction of conceptual knowledge on quality parameter (e.g. $COD_{filtered}$ is a subtype of COD).
However for the desired model functionality to reach the overall objective required for flow sheet design the class taxonomy is not sufficient alone. Similar to the problematic introduced in the section of class surrogates individuals play an essential role to formulate states of a wastewater quality. Again only through individuals relations between concepts can be expressed. Which is necessary to link fractionation rules and processes to quality parameter.
Eventually instances of concepts serve as delimitable fractions. This offers the possibility to express a state by a set of individuals and not as common by a list of state variables.
In contrast to individuals properties have an essential role in the material model to express relations between matter, oxygen and carbon equivalents. This model functionality is depicted in figure 4.12. Thereby the relation between the concept of VSS, particulate BOD_5 and BOD Ultimate (which eventually relates to biodegradable matter and oxygen equivalence) is shown. Through these relations changes through growth of organisms (consumption of substrate) is propagated to solids parameter. Vice versa changes on particulate matter (TSS, VSS, ISS) are propagated to parameter based on oxygen equivalence (BOD_5 and BOD_u). This essential functionality is not possible to describe by sole class taxonomy as shown in figure 4.11.
In contrast to figure 4.11, figure 4.12 shows only a cut out of the complete taxonomy. But additionally individuals are shown with the use of the `hasEquivalent` Relation.

4.5.4 Model interpretation and application

At the end of this section on the material model the question remains how the material model is eventually used to express a particular state of a wastewater stream. According to figure 4.3 the material model holds only generic knowledge on parameters for wastewater characterization. Hence this knowledge is case unspecific.
The JENA API forms the link between the model environment and the declarative model. Thereby entities (classes, individuals, and properties) in the environment map directly entities in the declarative model. In particular knowl-

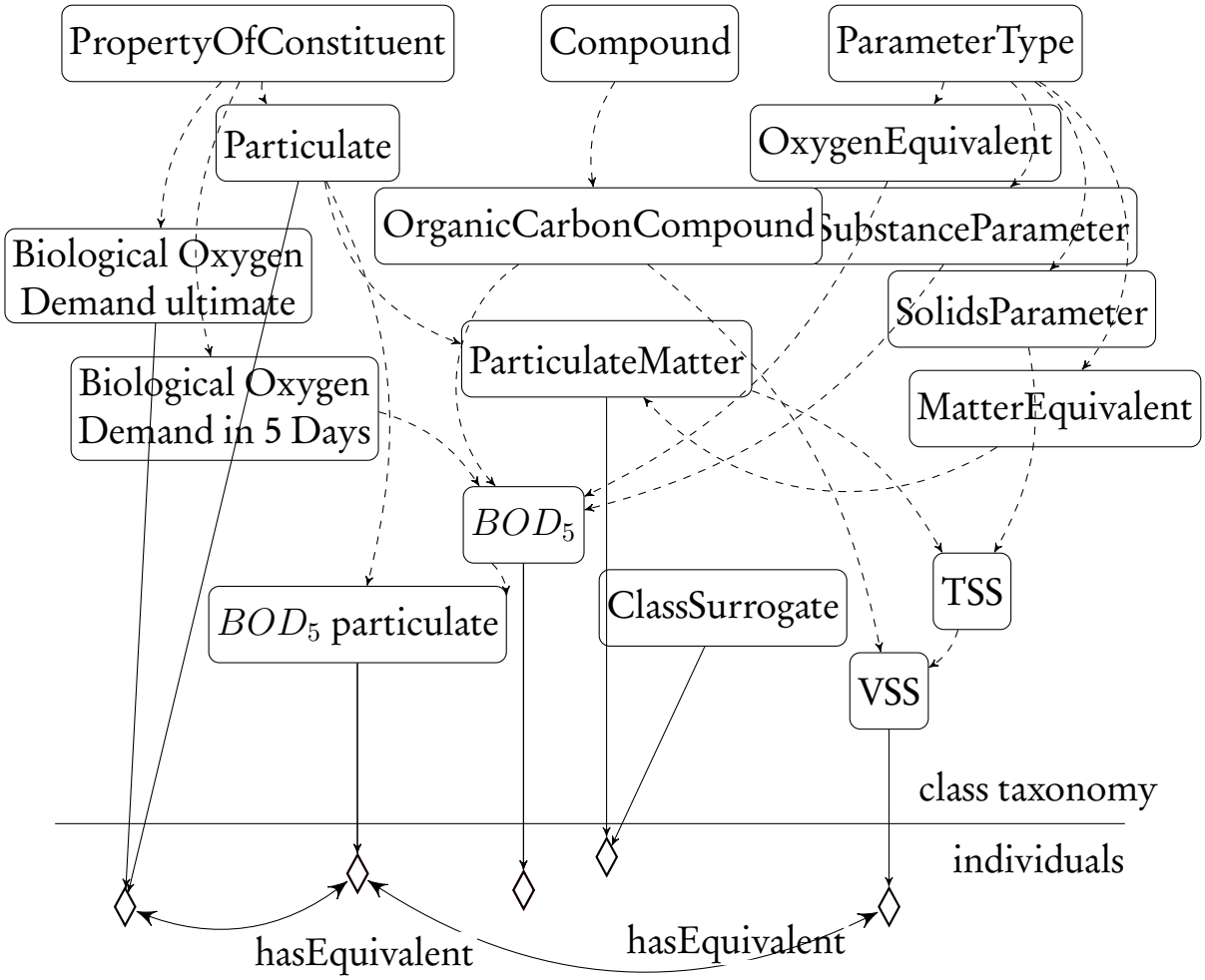


Figure 4.12: Relation between class taxonomy of concepts and instances. Classes are indicated by boxes above horizontal line. Class hierarchy is indicated by dashed lines. Individuals are indicated by diamonds below horizontal line. Membership of individuals to classes is indicted by solid lines from classes to individuals. Lines between individuals indicate property relation `hasEquivalent`.

edge on a single individual can be derived from the declarative model by JENA API methods.

There are two essential model functionalities regarding the modeling of a state. These are the extraction of a parameter value pair and the assignment or change of value of a parameter within a state. For example may the value of a variable such as COD be read or changed within a state due to a process. Within a conventional modeling environment the value of such a variable is simply read or changed. Within the approach presented in this work the value of a single variable is the sum of all individuals that share a respective property (e.g. being member of COD). The two functionalities of reading and changing a parameter value within a state are implemented in JAVA by the two methods GETAMOUNT and SETAMOUNT. This issue is also described in example 2.

Example 2 - Use of parameter COD by the declarative model
The representation of a particular state is shown in figure 4.13. In this example a state holds the only parameter value pair of COD = 100 mg/l. A single parameter value pair is modeled by the JAVA object QUANTITY. This object is defined by the two variables individual of type IND (JENA API) and amount of type DOUBLE. An overall state containing a set of quantities (qualitative and quantitative parameters) is modeled by the JAVA object QUANTITYSET. Figure 4.13 shows a prominent example of an initial state (before fractionation) where the total value of a qualitative parameter is assigned to a single individual which represents a class surrogate. After the fractionation the assigned value is distributed among a set of fractions (i.e. individuals) which assemble below the concept of `COD`. The class surrogate of COD itself is then not part of the QuantitySet anymore. In contrast to conventional modeling environments information on a single parameter is not stored within the modeling environment itself but is stored within the declarative model. Through queries the model environment can extract information on the individual. As in the example of figure 4.13 the environment may extract the information that the individual is member of class `COD`, `OxygenEquivalent`, etc.

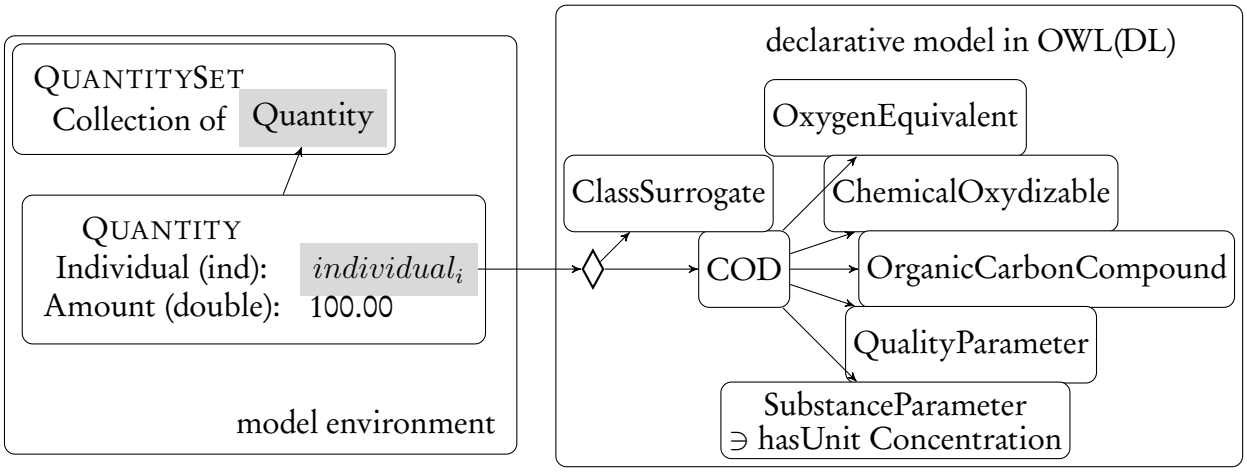


Figure 4.13: Interaction between model and model environment to represent the assignment between parameter and amount (e.g. COD=100 mg/l)

In listing 4.4 the method GETAMOUNT of class QuantitySet shows how a value of variable is read. Thereby two cases must be differentiated. In the first, the desired parameter is contained within the state and hence the asserted amount can directly be derived (e.g. asking for the amount of COD to the example shown in figure 4.13), line 4-6. Thereby the total concentration of 100.00 mg/l is solely attached to a single individual. In the second case, the value of a demanded parameter is distributed among a set of fractions. Then the amount of this parameter is the sum of all fractions that are subsumed by the class surrogate parameter, line 8-16.

Listing 4.5 shows the case of changing the value of a quantity within a state. Analogue to retrieving a parameter amount the case of a surrogate and a non-surrogate must be differentiated. Furthermore changes of parameter values that are related through the property `hasEquivalent` or is `isEquivalentOf` are also accounted for (c.f. section 4.5.3).

The method SETAMOUNT shown in listing 4.5 is applied when the value of a parameter is changed due to a process. If for example the amount of biodegradable matter is changed from 200 mg/l to 100 mg/l this value reduction must be applied to all fractions that share the property of being member of `BiodegradableMatter`. This is achieved by the code in line 16-33 of listing 4.5. Thereby the change of 50% reduction is propagated to all respective individuals. Additionally the property relation `hasEquivalent` is taken into account (line 29). Thereby values of concept `BOD5` are changed if values of `BODu` are changed (see figure 4.12). In case the parameter of which the value is to be changed is not a surrogate the amount of this parameter is directly altered (line 9-13). Line 7

Listing 4.4: Method GETAMOUNT of class QUANTITYSET

```
1   public double getAmount(Parameter parameter) {
2       double amount = 0.0;
3
4       if (contains(parameter)) {
5           return get(parameter).getAmount();
6       }
7
8       if (parameter.isSurrogate()) {
9           for (Quantity q : this) {
10              if (q.getParameter().isInstanceOf(
11                      parameter.getSurrogatedClass())) {
12                  amount += q.getAmount();
13              }
14          }
15          return amount;
16      }
17
18      return 0.0;
19  }
```

simply states that values of parameter can only be changed within a fractionated state (i.e. not accumulated).
In listing 4.4 and 4.5 the type INDIVIDUAL (as in figure 4.13) is represented by type PARAMETER which is a wrapper class that maps an individual which is a parameter.

4.6 FRACTIONATION AND ACCUMULATION OF STATES

4.6.1 Introduction

In the preceding sections fundamental concepts have been defined which are used to formulate the material concepts used within this work. Consequently any initial state s_i* can now be expressed as a set of individuals that are member of concept `QualityParameter`, `Compound` and `Substance`. However before a state can be applied every initial state must be fractionated.
Through the process of fractionation each individual that represents a quality parameters is substituted by a set of new individuals also referred to as fractions. Each of those substitutes represents not only the respective quality parameter but also a distinct list of properties of the original individual.
For an exemplary state the fractionation result is shown in table 4.7. In the left column a user specified state is listed being an outflow of a source. This initial state holds one quantity and two quality parameters (i.e. Ethanol, COD). After the fractionation process the amounts (i.e. concentration) of the two quality parameters are divided amongst related fractions (fractionated state)

Listing 4.5: Method SETAMOUNT of class QUANTITYSET

```
1  public void setAmount(Parameter parameter, double amount) {
2          if (amount < 0.0) {
3                  amount = 0.0;
4          }
5
6          if (isAccumulated())     return;
7
8          if (contains(parameter)) {
9                  double ratio = amount / get(parameter).getAmount();
10                 get(parameter).setAmount(amount);
11                 changeEquivalentParameter(parameter, amount, ratio);
12                 return;
13         }
14
15         if (parameter.isSurrogate()) {
16                 double amount_surrogate = getAmount(parameter);
17                 double ratio_surrogate;
18
19                 if (amount_surrogate == 0.0) {
20                         return;
21                 }
22                 ratio_surrogate = amount / amount_surrogate;
23
24                 for (Quantity q : this) {
25                         if (q.getParameter().isInstanceOf(
26                         parameter.getSurrogatedClass())) {
27                         q.setAmount(q.getAmount()    ratio_surrogate);
28                         changeEquivalentParameter(q.getParameter(), amount,
29                                                 ratio_surrogate);
30                         }
31                 }
32         }
33
34 }
```

according to fractionation rules (see table 4.8). From the fractionated state the quality parameters can be derived through accumulation (accumulated state).

The way a state is fractionated is defined through a list of fractionation rules (see section 4.6.4). An important model functionality regarding fractionation lies in the flexibility to choose an appropriate chain of fractionation rules depending on the data availability of quality parameters. Providing a minimum set of input parameters will lead to the use of more standard fractionation rules. Oppositely the provision of a comprehensive data set on input quality parameters will lead to the use of complex fractionation rules, reflecting better the true composition of a waste stream. Besides this flexibility the use of fractionation rules also allows for the recognition of overlapping contents of different quality parameters (e.g. relations between COD, BOD, Ethanol).

Each property in the declarative model is a value partition of disjoint subconcepts. For example may a single fraction represent particle size either by the assigned value dissolved or the value particulate (`ParticleSize` $\equiv$ `Dissolved` $\cup$ `Particulate`). Formal, if P stands for the property and V_1 and V_2 represent the only two values of P, than $P \equiv V_1 \cup V_2$ with $\varnothing = V_1 \cap V_2$ (V_1 and V_2 are disjoint, which means a member of P must either be member of V_1 or V_2). For a property P' with I possible disjoint values P' is defined by $P' \equiv \dot{\bigcup_{i \in I}} V_i$ with $V_i \cap V_j = \varnothing$ if $i \neq j$.

Table 4.7: Exemplary state before and after fractionation, based on fractionation rules as defined in table 4.8

Initial parameter set		**Fractionated state**		**Accumulated state**	
Parameter	Value	Fraction	Value	Parameter	Value
Q_h	1 m^3/h	Q_h	1 m^3/h	Q_h	1 m^3/h
Ethanol	500 g/m^3	iEthanol	500 g/m^3	Ethanol	500 g/m^3
COD	3000 g/m^3	iEthanolBOD	675 g/m^3	COD	3000 g/m^3
		iEthanolBODu	900 g/m^3	BOD	1480 g/m^3
		iCODiX	600 g/m^3	$COD_{filtered}$	1860 g/m^3
		iCODiS	600 g/m^3	$BOD_{filtered}$	997 g/m^3
		iBODuX	540 g/m^3	$BOD_{suspended}$	483 g/m^3
		iBODuS	360 g/m^3	$COD_{suspended}$	1140 g/m^3
		iBODS	322 g/m^3	TSS	760 g/m^3
		iBODX	483 g/m^3	VSS	760 g/m^3
		iVSSbio	360 g/m^3	COD/VSS	1,5
		iVSSunbio	400 g/m^3	Temperature	15
		COD/VSS	1,5		
		Temperature	15		

If QP is the set of quality parameters which is not empty ($QP \neq \varnothing$), and J is the number of properties P', than an unfractionated state is defined by the set of elements that fulfill proposition 4.16. Proposition 4.16 literally states that there are elements of QP which are not member of all properties.

$$\neg\forall x \left(x \in QP \rightarrow x \in \bigcap_{j \in J} P'_j \right) \tag{4.16}$$

After the fractionation proposition 4.17 must hold true. Proposition 4.17 literally says that all elements of QP x must also be element of every property.

$$\forall x \left(x \in QP \rightarrow x \in \bigcap_{j \in J} P'_j \right) \tag{4.17}$$

The above formal definition of states before and after fractionation are true only in a generalized context. In fact there are properties which need to be specified for all individuals. For example the property particle size must be specified for each fraction since by definition all wastewater constitutes prevail either in solid or dissolved form. Furthermore there may be properties that are represented by one value only, for example the property toxic. This property may have various sub-concepts that perhaps specify the degree of toxicity or type of affects, nevertheless there is no disjoint property such as nontoxic. For any fraction which is not allotted to be toxic it follows that there is either no information available regarding its toxicity or it is not toxic. In the scope of this work the latter option is applied. It follows for every one valued property the negation is true if the property itself is not specified.

4.6.2 Fractionation rules and algorithm

Through the process of fractionation the information content of a state is increased in terms of allocating properties to elements of this state. The knowledge required for fractioning is derived from: (i) expert knowledge; (ii) interdependencies between quality parameters; and (iii) predefined substance specific properties.
If an exemplary unfractionated state holds the parameter COD only, a predefined set of fractionation rules must be applied that state the breakdown according to the properties of particle size and biodegradability. There may be different rule sets for different types of wastewater (e.g. municipal wastewater, textile industry, etc.). The latter example refers to the incorporation of

expert knowledge. If the state holds the additional parameter $COD_{filtered}$ the particulate and dissolved fractions can be derived directly from the two parameters. This example refers to the accounting of interdependencies between quality parameter. Finally if a wastewater state holds the only quality parameter ethanol which at the same time stands for a particular substance, information on properties can be derived directly. In the case of ethanol the properties regarding particle size and biodegradability are intuitively specified within the knowledge base.

The process of fractionation is achieved through the execution of a list of so called fractionation rules. A single fractionation rule has a number of postconditions, preconditions, and a list of allocation terms that eventually define how the values of input fractions is allocated to output fractions of this rule. Preconditions are divided into the two sub properties `require` and `absent`. Thereby the property `require` specifies which parameter must be present within a state for the rule to be applied. Oppositely the property `absent` specifies which parameter must be absent within a state for a rule to be applied. The postcondition `decrease` specifies which fraction will be removed from the state after a rule has been applied. Finally the value allocation from the input fractions to the output fractions is attached to a single fractionation rule via the property `hasAccount`.

The class definition of concept `FractionationRule` is depicted in proposition 4.18.

$$\begin{aligned} \texttt{FractionationRule} \equiv\ & \forall \texttt{absent.QualityParameter} \sqcap \\ & \forall \texttt{require.QualityParamter} \sqcap \\ & \forall \texttt{decrease.QualityParameter} \sqcap \\ & \forall \texttt{hasAccount.Account} \sqcap \exists \texttt{hasAccount.Account} \end{aligned} \tag{4.18}$$

The fractionation algorithm is straight forward in that sense that fractionation rules are applied to a state (i.e. set of fractions) as long as there exists at least one fractionation rule whose preconditions are fulfilled. If F stands for the set of elements within a state and R is the set of possible fractionation rules and $V()$ is a function that returns the validity of all preconditions of a rule than the algorithm is repeated as long $\exists x \exists y\,(x \in R \wedge y \in F \wedge V(x,y))$. The

fractionation is terminated if the following proposition holds true:

$$\forall x \forall y \left((x \in R \wedge y \in F) \rightarrow \neg V(x, y) \right).$$

The practical implementation of the fractionation algorithm is shown in listing 4.6. In general the fractionation loop is processed as long as there are fractionation rules that can be applied on a given state (i.e. QUANTITYSET). Therefore all fractionation rules are checked according to their preconditions. In listing 4.6 lines 4 to 17 all rules are identified where the state fulfills the condition of required, to be decreased or absent parameters is met. If the thereby identified list of candidates is empty the fractionation is aborted. If not the candidate list is sorted according to the fractionation strategy. The fractionation strategy thereby determines the ranking of a number of fractionation rules. The applied strategy privileges the rule with the least number of preconditions (line 21 to 23). Eventually the winning fractionation rule is applied (line 35). Additional parameters that have been removed from the state as result from fractionation are memorized in an attic list for subsequent fractionation steps.

4.6.3 Accumulation

In contrast to the step of fractionation which is applied once for every initial state accumulation is applied after each state transition. This is necessary to derive the value of primary parameter. The fundamental aspect of accumulation lies in the extraction of a parameter value from a given set of fractions which is already described in section 4.5.4.

4.6.4 Exemplary fractionation

Table 4.7 shows the results of an exemplary fractionation. In the left column the parameter of a initial state are listed by the parameter of COD, Ethanol and the flow rate of 1 m^3/h. Thereby the two quality parameters are defined as primary parameter and are represented by individuals of concept surrogate. According to the formal definition of fractionation the state is unfractionated because these two individuals are not member of any necessary property concept. The same initial state is shown In figure 4.14 regarding the relation between individuals and concepts in the ontology.
Table 4.8 lists a standard rule set of nine fractionation rules. The application of this rule set to the exemplary state results in the fractionated state in table 4.7 center column. The relation between individuals and concepts of the ontology regarding the fractionated state are shown in figure 4.15.

Listing 4.6: Fractionation algorithm (method FRACTIONATION of class FRACTIONATION)

```
1  public boolean fraction() {
2          List<FractionationRule> candidates =
3                                  new ArrayList<FractionationRule>();
4
5          ruleLoop: for (FractionationRule rule : FractionationRule
6                          .getAllFractioningRules()) {
7
8                  for (Parameter param : rule.getRequires())
9                          if (get(param, true) == null)
10                                 continue ruleLoop;
11                 for (Parameter param : rule.getDecreases())
12                         if (get(param, false) == null)
13                                 continue ruleLoop;
14                 for (Parameter param : rule.getAbsent())
15                         if (get(param, false) != null)
16                                 continue ruleLoop;
17                 candidates.add(rule);
18         }
19
20         if (candidates.isEmpty())        return false;
21
22         Collections.sort(candidates, strategy);
23
24         FractionationRule winner = candidates.get(0);
25
26         QuantitySet gatheredQty = new QuantitySet();
27         for (Parameter p : winner.getRequires()) {
28                 gatheredQty.add(get(p, true));
29         }
30         for (Parameter p : winner.getDecreases()) {
31                 Quantity q = get(p, false);
32                 if (!gatheredQty.contains(p))
33                         gatheredQty.add(q);
34         }
35
36         winner.transform(contents, gatheredQty);
37
38         for (Parameter p : winner.getDecreases()) {
39                 contents.remove(gatheredQty.get(p));
40                 attic.add(gatheredQty.get(p));
41         }
42
43         return true;
44 }
```

The applied rule chain for the given example state is as follows: 9-2-1-3-4-5-6-7-8, according to the given rule numbers in table 4.8. The rule with the least conditions is rule number 9. With this rule a value for temperature is set if this is not specified.
In the following step all organic matter which will contribute to concept of oxygen equivalence is translated to fractions that also assemble under the concept oxygen equivalence. In the example Ethanol (which is a matter equivalence) obtains a double under the concept BOD in five days and ultimate (rule 1 and 2). By the next applied rule (#3) the link to solids parameter is established since the initial state contains no solids parameter by itself. Due to the absence of any further parameter besides COD in the family of oxygen equivalence a intermediate fraction BOD_{Rest} is formed (rule # 4). Through this intermediate fractions the portion of biodegradable oxygen demand introduced by parameter of matter equivalence (Ethanol) is corrected. Rule 4 also shows the advantage of the material model to use class surrogates within algebraic terms instead of single variables. By rule 6 and 7 the distribution between particulate and dissolved biodegradable fractions are defined though standard values (since the initial state itself provides no information in this matter). The constants in table 4.8 f_{ubs}, f_{fubx}, f_{eb} and k are taken from Henze *et al.* (2008, p. 58-59).
In table 4.7 left column the accumulated state is shown. It holds all parameter of the initial state plus any other quality parameter which can be derived from the fractionated state.

4.7 Process model

4.7.1 Introduction

The material concept is applied for representing states. As counterpart, tasks in the perspective of STN are represented by the concept of `ModelUnit`. A model unit represents therefore any device or reactor wherein a state can be changed through processes.
Concluding from the literature review on process models there are two contrary approaches in process modeling, phenomenological process modeling and the conventional flow sheet based approach. In the first no process representations are directly linked to devices. Instead occurring processes are derived solely from (i) the present substances and (ii) the present context within a device (e.g. pressure, temperature). This approach requires some sort of a material model which defines the relations and reactions between substances depending on the surrounding context (Linninger *et al.*, 1996b; Linninger and

Table 4.8: Exemplary rule set involved in fractionating parameter list according to table 4.7

#	Precondition		Postcondition	Account
	absent	require	remove	
1	iEthanolBOD	iEthanol		$iEthanolBOD \leftarrow iEthanol \cdot 1.35$
2	iEthanolBODu	iEthanol		$iEthanolBODu \leftarrow iEthanol \cdot 1.8$
3	iCODtoVSS, iVSS	iCODtotal		$iCODtoVSS \leftarrow 1.5$
4	iBOD, iBODfiltered, iCODtotal	iCODtotal	iCODtotal	$iBODRest \leftarrow csBiodegradableOxygenDemand - (iCODtotal \cdot (1 - f_{ubs} + f_{ubx}))$; $iCODi \leftarrow csBiodegradableOxygenDemand - (iCODtotal \cdot (f_{ubs} + f_{ubx}))$
5		iCODi	iCODi	$iCODiS \leftarrow iCODi \cdot (f_{ubs}/(f_{ubs} + f_{ubx}))$ $iCODiX \leftarrow iCODi \cdot (f_{ubx}/(f_{ubs} + f_{ubx}))$
6		iBODuRest	iBODuRest	$iBODuX \leftarrow iBODuRest \cdot (1 - f_{eb})$ $iBODuS \leftarrow iBODuRest \cdot f_{eb}$
7	iBODS, iBODX	iBODuS, iBODuX		$iBODS \leftarrow iBODuS \cdot (1 - exp(-5 \cdot k))$ $iBODX \leftarrow iBODuX \cdot (1 - exp(-5 \cdot k))$
8	iVSSunbio, iVSSbio	iCODiX, iBODuX, iCOD-toVSS		$iVSSbio \leftarrow iBODuX/iCODtoVSS$ $iVSSunbio \leftarrow iCODiX/iCODtoVSS$
9	iTemperature			$iTemperature \leftarrow 15$

Stephanopoulos, 1998; Marquardt *et al.*, 2010; Morbach *et al.*, 2008d; Yang *et al.*, 2003).

In this work a material model has been developed which defines relationships between substances. But the material model does not directly link substances with context and occurring reactions. Hence in this work a conventional flow sheet based approach (see section 2.5.7.3) has been applied. Therefore processes are attributed directly to devices.

There are two reasons for this design decision. First, the importance of boundary conditions such as temperature or pressures has been placed little value on, although it can not totally be neglected. Second, an important design criterion was the inclusion of biological processes with focus on carbon removal.

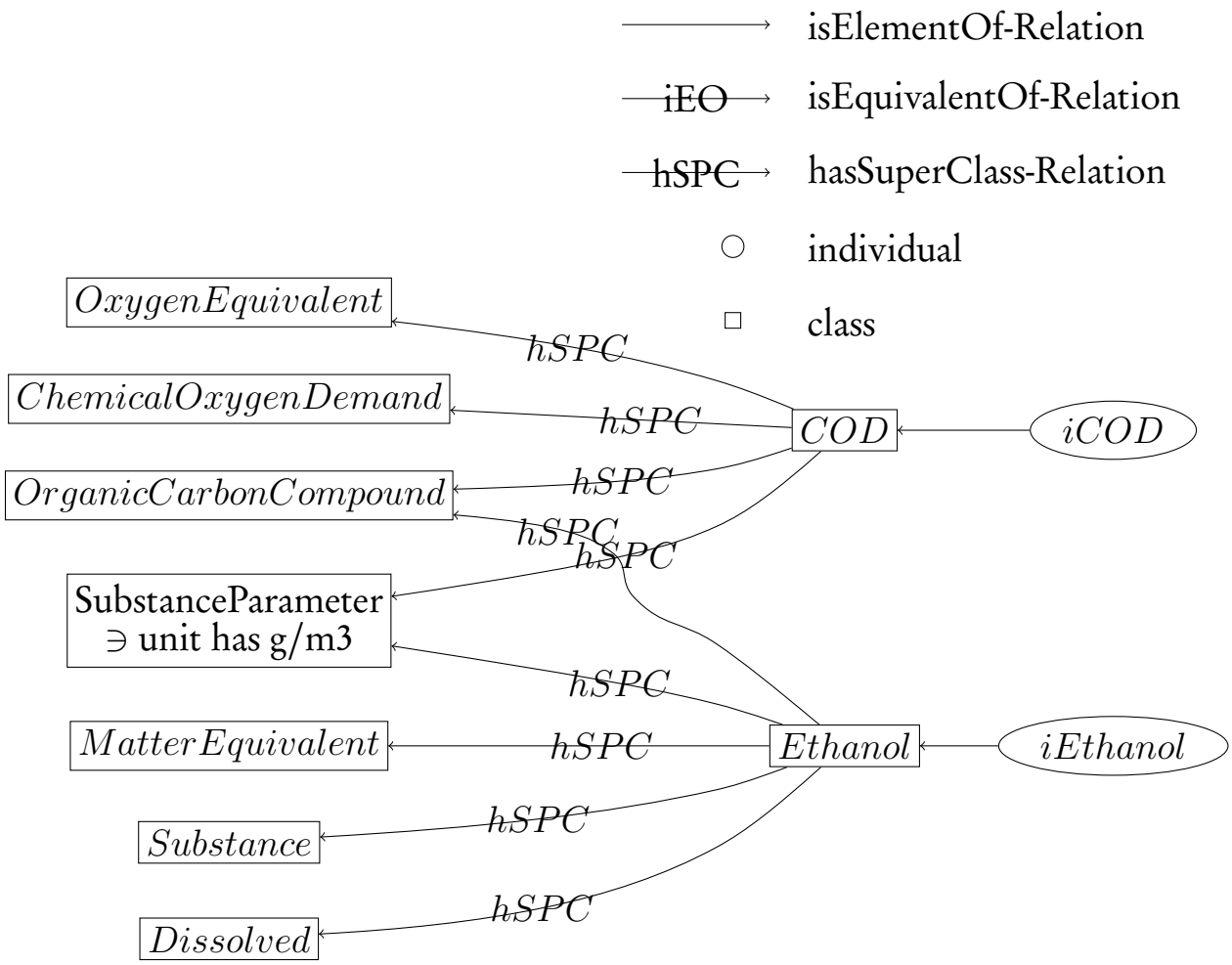


Figure 4.14: Individuals in initial state according to table 4.7 (only quality parameter indicated)

In contrast to chemical reactions with known educts and products, the formulation of biological processes is more complex. In particular the description of biological processes requires a set of assumptions which are case dependent. Hence processes have been attributed to devices to allow for a generally valid material model.

Every model unit is not only described through a list of processes but may also have a list of preconditions assigned to. Thereby preconditions are used to specify under which boundary conditions a particular device can be applied. Furthermore may a model unit be described by cost functions as well as other properties such as volume or surface area. Used process descriptions are solely based on process models taken from literature.

The concept `ModelUnit` builds up on the fundamental model fragments `Term`, `Condition`, `Process`, and `Port`. The `Port` concept is simply introduced to ease implementation and comprehension. A single individual of concept `ModelUnit` is assembled as shown in figure 4.16. The in- and outlet

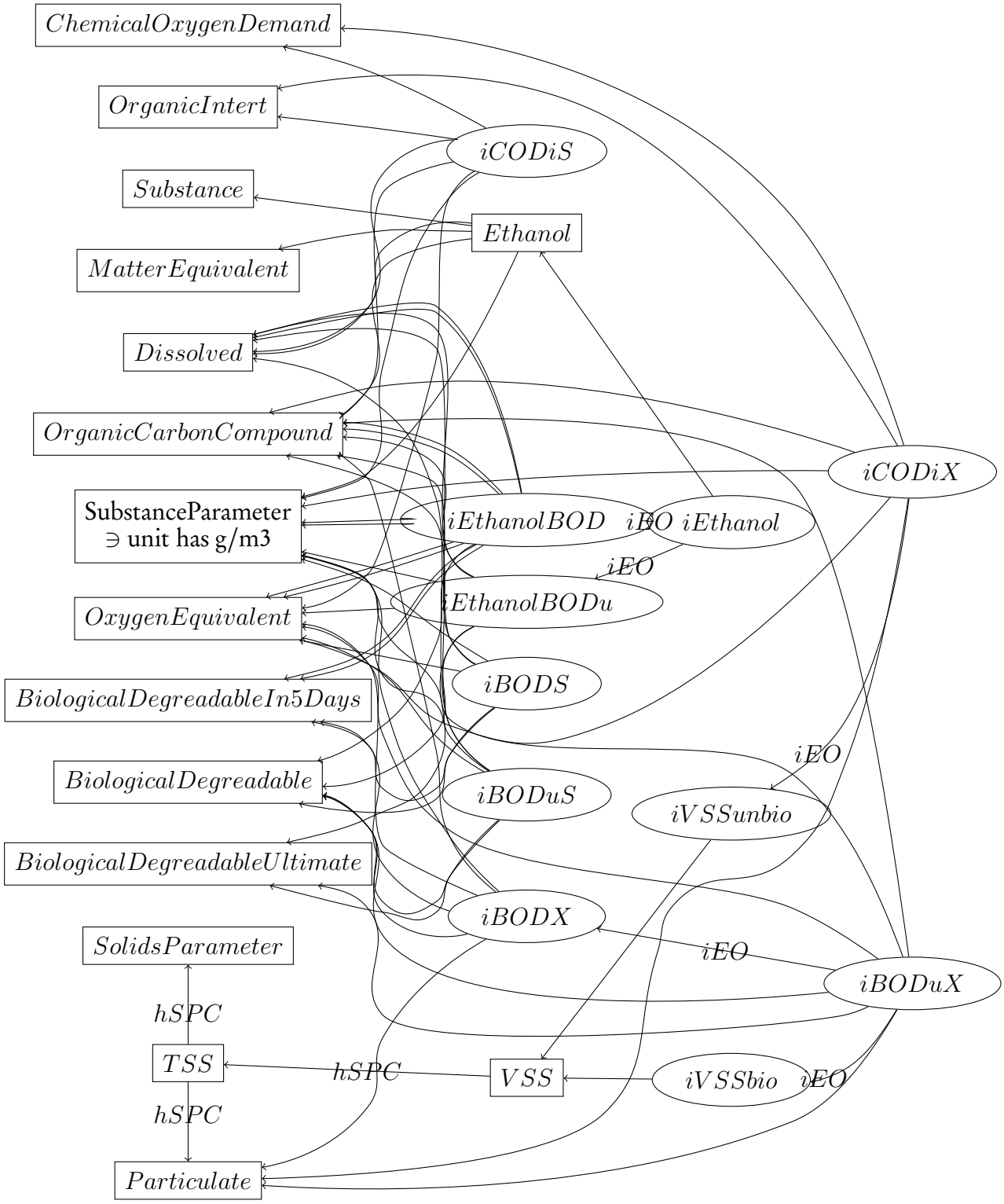


Figure 4.15: Individuals in fractionated state according to table 4.7; only quality parameter indicated (for legend see figure 4.14)

ports are attached to a single model unit via the properties `hasInletPort`, `hasPrimaryOutletPort`, and `hasSecondaryOutletPort` (in figure 4.16 indicated by hIP and hOP). The number of inlet and outlet ports depends on the type of model unit (see next section). Every inlet port may have an arbitrary

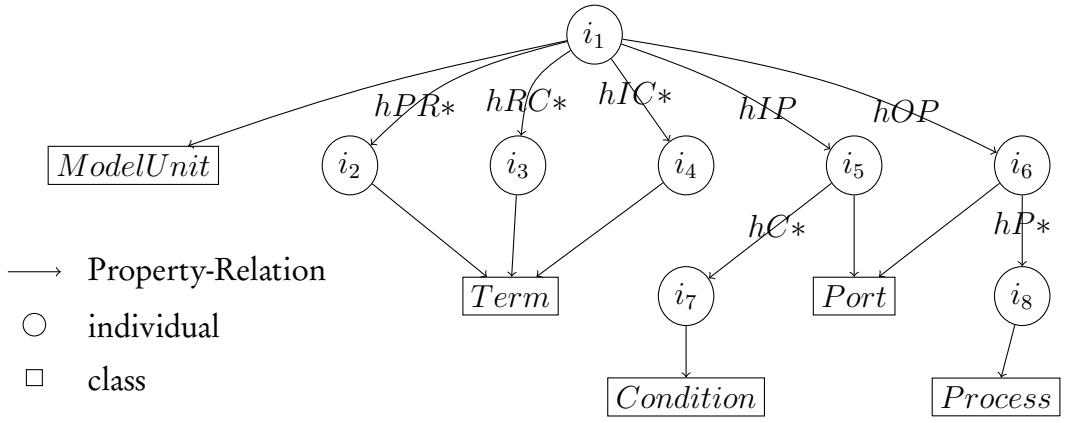


Figure 4.16: Graph representation of the `ModelUnit` concept; *more than one relation possible

number of conditions, connected via the property `hasCondition` (hIC). Every outlet port may have an arbitrary number of processes (see section 4.4.6), connected via the property `hasProcess` (hP).

4.7.2 Generic model units

In view of the search algorithm to eventually identify feasible treatment chains rigid formalization of model unit types is necessary. The minimal number of models unit types used is depicted in figure 4.17. The six generic model unit types differ mainly with regard to assigned ports and functionality.
Furthermore, the set of units may be divided by groups of two. The first group enclosing the types source and sink unit are special in that kind that they are not used for state transition. A source unit has only one outlet port with no processes attached. Every source unit bears a user specified initial state* which in turn is fractionated before the state is emitted by the outlet port. In contrast a sink unit has only one inlet port with an arbitrary number of preconditions. Similar to source units are those preconditions user specified for each application. Summarizing, source and sink units form the start and end nodes of each planning graph. Source and sink units are the only types of generic model units which are specified by the user to represent a particular design question.

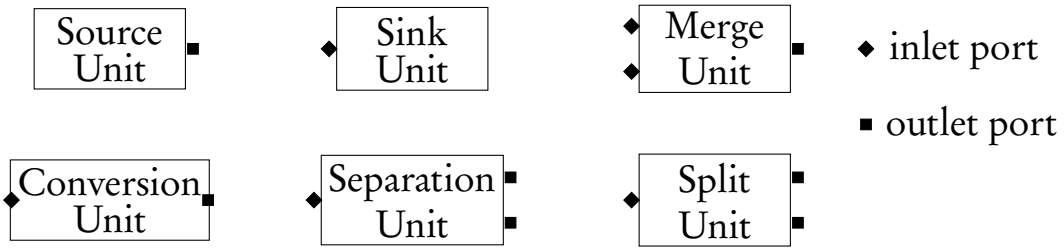


Figure 4.17: Generic unit blocks with inlet and outlet ports; Outlet ports of separation and sink units vary regarding assigned hydraulic status (see table 4.9)

The second group formed by the merge and split unit are of truly universal nature. This is because they are described only once but can be included within a plan in arbitrary number. Thereby the merge unit has two inlet ports with no preconditions as well as one outlet port with no processes attached. Through the merge unit two incoming states are merged into one by a simple allegation alternate. Since a state holds the concentrations of fractions, merging brings about no difficulties. By a sequence of one to n merge units an arbitrary number of states can be merged. As counterpart a split unit divides an incoming state into two outgoing states. Thereby the outgoing states differ only with regard to flow rate as specified by the incoming state.
Any technological device for wastewater treatment or conditioning may be specified by one of the two left generic model types, conversion unit and separation unit. Hence the eventual knowledge base on treatment technologies holds an arbitrary number of specifications of those two types of units. Any conversion unit has one inlet port with preconditions as well as one outlet port with processes. In contrast a separation unit has two outlet ports. In contrast to source and sink units, conversion and separation units may not be altered for each application but are selected from the knowledge base and applied only if preconditions permit the particular use.
With the above listed six generic model units it is possible to form a cyclic graph. Ideally a new treatment chain can be generated by subsequently adding new units to an existing chain. Applying such a forward planning algorithm already incorporate units can not be replaced or modified. In contrast possible candidate units are always chosen as a result of the state transitions of the already included units.

4.7.3 Postconditions and Preconditions

The main feature of the introduced `ModelUnit` concept is the representation of processes that are related to this unit. The representation of process models is essential in environmental modeling but resulting from the layered model representation processes can be further described according to a context of use. Most important each `ModelUnit` concept can be described by so called preconditions. Thereby preconditions must be fulfilled before a model unit can be added to a plan.
Within the `ModelUnit` concept preconditions are attached to a port individual (i.e. inlet port). The precondition itself is based on the `Condition` concept and is related to the port individual via the property `hasCondition` (hC) (Figure 4.16).

Postconditions of model units differ to preconditions in that sense that they qualitatively describe the benefit of applying a particular model unit. Postconditions are not defined in the declarative model. Instead the postcondition of each unit is processed during runtime of the modeling environment whenever it is required. Thereby an incoming state to a unit is processed and the difference between incoming and outcoming state is used to derive the postcondition of the respective model unit. A single postcondition is defined by the Java class GOAL which is set up of a parameter and a qualitative value {decrease, increase, change} (e.g. decrease of biodegradable matter). An example of the use of postconditions is also given in table 4.15.

4.7.4 Recycle flow

An important model functionality regarding the conceptual design of wastewater treatment flow sheet is the ability to sequentially add model units to a given initial plan to eventually obtain a complete plan. This corresponds to the applied forward search as described in sections 4.8. However from a practical point of view such a strict sequential design approach is not always reasonable. Because this forwarding allows only forward information transfer between a successor and predecessor unit. A prominent example of such a problem is the design of biological carbon removal using an MBR with adjacent separation of activated sludge flocs applying a secondary settler or flotation tank for recycle of active microorganisms. In practice, dimensioning of aeration tank and adjacent settler can not be recognized separately.

This issue has been addressed by the introduction of two other quantitative parameters for recycle and waste flow, Q_r and Q_w. Together with the parameter for hourly flow Q_h these parameters are used within an index for hydraulic status (hs), see table 4.9. The hydraulic status is used to internally define a so called secondary precondition to control the formation of cycles as specified in table 4.9.

The application of the hydraulic status for the formation of cycles is further described in section 4.8.3 and figure 4.20. This approach still does not allow for an iteratively design of MBR systems, but information on recycle and waste flow can be propagated in the search process to be able to recognize and close cycles.

Table 4.9: Definition of hydraulic status (hs)

Hydraulic Status (hs)	definition
0	state contains only Q_h
1	state contains Q_h, Q_r, Q_w with $Q_h > Q_r + Q_w$
2	state contains Q_h, Q_r, Q_w with $Q_h = Q_r + Q_w$
3	state contains Q_h, Q_r with $Q_h = Q_r$
4	any other condition → failure

Table 4.10: Secondary preconditions of unit types

Unit type	Allowed hs value	Comment
primary inlet port sink unit	0	
primary inlet port merge unit	0	
secondary inlet port merge unit	0 or 3	hs=3 is required to close a cycle
primary inlet port conversion unit	0 or 1	hs=1 allows to apply conversion units in a series
primary inlet port settler	0 or 1	hs=0 allows for the definition of primary settlers (such a unit must have a definition for the separation of over and under flow)
primary inlet port split unit	2	

4.7.5 Biological carbon removal (with MBR)

As introduced in the section before, the consideration of biological processes in particular with respect to carbon removal play an important role in wastewater treatment and hence need particular modeling attention[5].
Essential design criteria for the use of aerated or not aerated tanks in MBR systems are the dimensioning of the reactor volume as well as the definition of recycle and waste flow (excess sludge). These aspects are independent of the applied process models for conversion and separation units.
For all units, including MBR systems, the model unit concept allows for (i) a predefined dimensioning or (ii) for dimensioning according to the inflowing state. In the first case units need a rigid formulation of preconditions to guarantee the intended process model application and its practical feasibility. For example, an aeration tank with a fixed volume may only allow for a limited

[5]The dilemma in this work regarding modeling of conversion processes for carbon removal lies in the choice between simple and complex models. Simple static models would be appropriate regarding the low detail of data. In contrast the use of fractions as facilitated by the material model suggest the use of complex model (e.g. ASM like approaches). But complex models require the use of reaction parameters which are commonly unknown without detailed wastewater analysis.
As stated in section 4.4.6 the decision has been made to apply complex models using standard values for reaction parameters.

inflow volume with limited load of biodegradable matter. In the second case the volume, recycle and waste flow of an aeration tank are defined according to the inflow stream and load of biodegradable matter.
All three design parameters (volume, waste flow, recycle ratio) can be specified via the properties shown in proposition 4.19 to 4.21.

```
modelUnit    hasReactorVolume    term        (4.19)
modelUnit      hasWasteFlow      term        (4.20)
modelUnit    hasRecycleRatio     term        (4.21)
modelUnit      hasUnderflow      term        (4.22)
```

A short guide on volume dimensioning of aeration tanks for carbon removal according to Henze *et al.* (2008) is given in appendix A.

4.7.6 Exemplary model units

After theoretical aspects about model units have been discussed in the above sections, this section shows how particular modeling units can be defined. From the six generic model unit types (see figure 4.17) only conversion and settling units can be defined within the knowledge base. Besides sink and source units can be defined with regard to limit values and initial wastewater characteristics.
Aspects that must be defined for conversion and sink units are preconditions, processes and additional properties. Additional properties can be cost information, recycle ratio, waste flow, and reactor volume. Cost information are separated into running cost and investment cost as specified by the two triples as shown in proposition 4.23 and 4.24.

```
modelUnit    hasRunningCost    term        (4.23)
modelUnit     hasInvestCost    term        (4.24)
```

Preconditions are not directly linked to a model unit but as shown in figure 4.16 are linked to the inlet port of a model unit. Thereby preconditions are based on the `Condition` term.
Table 4.12 shows five conversion units which have been defined within the OWL based knowledge base. Conversion unit #1 and #2 are two examples of a conventional aeration reactor for carbon removal. Thereby conversion unit #2 has a fixed reactor volume of 100 m^3 and conversion unit #1 uses a function for volume dimensioning as specified in equation A.9. The waste flow

(i.e. excess sludge), Q_w of both reactors is defined by equation A.11. The recycle ratio (R) is defined by a constant by the values of 1 and 0.8 (i.e. 100% and 80%). The precondition of both conversion unit #1 and #2 are equal as defined in table 4.12. As an advantage of the material model the left side of a condition can be a particular substance (e.g. Ammonia) or a conceptual parameter such as biodegradable soluble matter, which is represented as a class surrogate. The actual conversion processes are linked analogue to preconditions not directly to the unit itself but to the corresponding outlet port of a unit. Through this a particular outlet or inlet port can be reused by different units. In other words a number of units can share the same outlet or inlet ports. In the case of conversion unit #1 and #2 the linked processes are aerobic growth of heterotrophic biomass, decay of heterotrophic biomass and hydrolysis as specified in table A.3.

The implementation of the process of growth of heterotrophic biomass for conversion unit #1 and #2 differs from the specification in table A.3 as it has no stoichiometric term for ammonia.

The specifications of cost regarding the different treatment units in table 4.11 and4.12 are solely of exemplary character. The defined costs only show the possibility to define running and investment cost as function of wastewater and model unit characteristics. As example investment cost may be specified as function of reactor volume and running cost may be specified as function of load of biodegradable matter.

Conversion unit #3 and #4 define reactors for nitrification and denitrification. Both with fixed reactor volume, waste flow and recycle ratio. For a serious application these properties would have to be defined by appropriate dimensioning rules.

Conversion units #1 to #4 are based on the MBR system which requires the definition of a recycle ratio as well as the definition of the amount of excess sludge. In contrast conversion unit #5 describes a fermentation unit with fixed biomass. There by the processes are implemented as described in table A.4 with a fixed concentration of biomass. At present it is not possible to represent a true UASB reactor by the defined generic model units since conversion units allow only for conversion processes and separation units allow only for separation of substances. To overcome this shortage a further generic model unit would be required which allows for the incorporation of both conversion processes and separation processes.

Table 4.11: Overview on exemplary model units - Separation units

Properties	Preconditions	Processes	Cost
separation unit #1 - settler			
$Q_{underflow} \leftarrow 0.01 \cdot Q_h$	Particulate Matter $> 20 \quad g/m^3$	Removal efficiency of Particulate Matter $\eta = 1 - 0,15$	-
separation unit #1 - secondary settler			
	Particulate Matter $> 20 \quad g/m^3$	Removal efficiency of Particulate Matter $\eta = 1 - 0,03$	$C_{invest} = 200$

Table 4.11 shows two separation units. Thereby separation unit #1 is intended to represent a primary settler with a fixed underflow and a fixed removal efficiency on particulate matter of 85 %. Separation unit #2 represents a secondary settler with a fixed removal efficiency of 97 % of particulate matter. The overflow and underflow rate is not defined since is used from the proceeding conversion unit (see therefore section 4.7.4 on recycle flow).
The definition of all conversion and separation units as shown in table 4.12 and 4.11 have been transcripted into OWL(DL) as specified in figure 4.16. For the example of conversion unit #4 the OWL(DL) code is shown in listing 4.7. Thereby the definition of properties such as recycle ratio, waste flow and reactor volume are defined in lines 5 to 7. The inlet and outlet port is defined in lines 8 and 9.

Listing 4.7: Encoding of nitrification unit in OWL(DL)

```
<owl:NamedIndividual rdf:about="#ModelUnit_Nitrification_1">
    <rdf:type rdf:resource="#ConversionUnit"/>
    <rdfs:label xml:lang="en">nitrification (1)</rdfs:label>
    <hasModelUnitProperty_RecycleRatio rdf:resource="#Constant_1"/>
    <hasModelUnitProperty_ReactorVolume rdf:resource="#Constant_100"/>
    <hasModelUnitProperty_WasteFlow rdf:resource="#Constant_200"/>
    <hasPrimaryInletPort rdf:resource="#InletPort_10"/>
    <hasPrimaryOutletPort rdf:resource="#PrimaryOutletPort_11"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#PrimaryOutletPort_11">
    <rdf:type rdf:resource="#PrimaryOutletPort"/>
    <isPrimaryOutletPort rdf:resource="#ModelUnit_Nitrification_1"/>
    <hasProcess rdf:resource="#Process_Decay_Autotrophs_1"/>
    <hasProcess rdf:resource="#Process_Decay_Heterotrophs_2"/>
    <hasProcess rdf:resource="#Process_Growth_Aerobic_Autotrophs"/>
    <hasProcess rdf:resource="#Process_Growth_Aerobic_Heterotrophic_2"/>
    <hasProcess rdf:resource="#Process_Hydrolysis"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#InletPort_10">
    <rdf:type rdf:resource="#InletPort"/>
    <hasInputCondition rdf:resource="#Condition_37"/>
    <hasInputCondition rdf:resource="#Condition_71"/>
    <isPrimaryInletPort rdf:resource="#ModelUnit_Nitrification_1"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#Condition_71">
    <rdf:type rdf:resource="#Condition"/>
    <rdfs:label xml:lang="en">Condition: NH4&amp;gt;1</rdfs:label>
    <hasRightSide rdf:resource="#Constant_1"/>
    <isInputConditionOf rdf:resource="#InletPort_80"/>
    <hasComparator rdf:resource="#LargerThan"/>
    <hasLeftSide rdf:resource="#iAmmonia"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#Constant_1">
    <rdf:type rdf:resource="#Constant"/>
<factor rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</factor>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#Process_Decay_Autotrophs_1">
    <rdf:type rdf:resource="#Process"/>
    <rdfs:label xml:lang="en">Process: decay of autotrophs #1</rdfs:label>
    <hasStoichiometricTerm rdf:resource="#Assign_59"/>
    <hasStoichiometricTerm rdf:resource="#Assign_61"/>
    <hasStoichiometricTerm rdf:resource="#Assign_64"/>
    <hasRate rdf:resource="#Multiply_5"/>
</owl:NamedIndividual>
```

Table 4.12: Overview on exemplary model units - Conversion units

Properties	**Preconditions**	**Processes** see table A.3 and A.4	**Cost**
conversion unit #1 - carbon removal (aerobic reactor)			
$R = 1$ $Q_w \rightarrow$ see eq. A.11 $V \rightarrow$see eq. A.9	$Q_h > 4 \quad m^3/h$ Biodegradable Soluble Matter $> 50\, g/m^3$ Biodegradable Particulate Matter < 200 g/m^3 $NH4 < 10\, g/m^3$	aerobic growth of heterotrophs decay of heterotrophs hydrolysis	$C_{invest} = V \cdot 25$ $C_{running} = L_{BioDeg} \cdot 0.8$
conversion unit #2 - carbon removal (aerobic reactor)			
$R = 0.8$ $Q_w \rightarrow$ see eq. A.11 $V = 200\, m^3$	$Q_h > 4 \quad m^3/h$ Biodegradable Soluble Matter $> 50\, g/m^3$ Biodegradable Particulate Matter < 200 g/m^3 $NH4 < 10\, g/m^3$	aerobic growth of heterotrophs decay of heterotrophs hydrolysis	$C_{invest} = 5000$ $C_{running} = L_{BioDeg} \cdot 0.8$
conversion unit #3 - denitrification			
$R = 1$ $Q_w = 0.1\, m^3/h$ $V = 100\, m^3$	$NH_4 > 1\, g/m^3$ Nitrate $> 1\, g/m^3$ Biodegradable Soluble Matter $> 50\, g/m^3$ Biodegradable Particulate Matter < 200 g/m^3	anoxic growth of heterotrophs decay of heterotrophs decay of autotrophs hydrolysis	-
conversion unit #4 - nitrification			
$R = 1$ $Q_w = 0.2\, m^3/h$ $V = 100\, m^3$	$NH_4 > 1\, g/m^3$ Biodegradable Soluble Matter $> 50\, g/m^3$ Biodegradable Particulate Matter < 200 g/m^3	aerobic growth of heterotrophs aerobic growth of autotrophs decay of heterotrophs decay of autotrophs hydrolysis	-
conversion unit #5 - fermentation			
$V = 100\, m^3$	active biomass $< 1\, g/m^3$	fermentation hydrolysis	$C_{invest} = 200$

4.8 Flow sheet generation

4.8.1 Introduction

This sections describes how the objective to generate flow sheets for wastewater treatment is deployed in this work. In a generalized way this objective is described as a planning problem in the context of Artificial Intelligence (AI). Thereby a planning problem is concerned with finding the path (i.e. sequence of state transitions) between a given initial state to a desired goal state (Russel and Norvig, 2003, p. 465).
The key in solving a planning problem lies essentially in the representation of the problem itself. The focus thereby lies on (Linninger *et al.*, 2000a; Russel and Norvig, 2003; Weld, 1999): (i) representation of states; (ii) representation of actions (syn. task, state transition); and (iii) representation of a goal.
A state in the context of flow sheet generation is any plan resembling a set of treatment units and states of waste streams. To distinguish between the meaning of a state as a condition of a waste stream and state as a plan the latter is indicated by a * as state* from now on. Three types of states* can be identified. First, the initial state* (or initial plan) sets the task of the planning problem. Second, the goal state* (or finial state) is to be identified as result of the planning problem. Any intermediate state* on the path between initial to final state belongs to the third type of incomplete plans.
The representation of actions defines how a state* can be transformed into a successor state* possibly moving closer to the desired goal state*. Each action holds a logical component as well as a execution component. The logical component is set up by preconditions and postconditions. Through preconditions the applicability of an action with respect to the state to be transformed is defined. Thereby preconditions define the scope of application for each action. In case of goal based search (e.g. MEA) the logical component has additionally a set of postconditions. Thereby postconditions qualitatively describe what goal can be achieved by the application of the respective action. The execution component eventually generates the successor state through defined processes. In the case of this work an action to be applied to generate a new state* relates to a new model unit added to a state*. the logical and execution component are not part of the action but of the inserted model unit.
As third aspect besides the representation of states* and actions is the definition of the final goal. In the context of flow sheet generation the final goal states* are unknown. But any goal state* must fulfill the properties of being complete and correct. Thereby a complete plan is defined as any plan which holds no

Table 4.13: Term definition regarding flow sheet generation

Term	Definition
plan	a set of units and wastewater states, regarding state space search a plan is also referred to as state*
initial plan	a set of source and sink units (syn. initial state*)
correct plan	a plan where all preconditions of included units are fulfilled; in turn an incorrect plan holds units whose preconditions are violated;
complete plan	a plan which holds no open states or units (can be represented as a graph where all nodes are linked by edges to other nodes)
state	a set of parameter value pairs that defines the condition of a wastewater stream
open state	any state within a plan which has no successor unit is referred to as open state
open unit	any unit within a plan which has at least one open inlet port is referred to as open unit
state*	in the context of state space search a state* referrers to any possible condition that can be reached from the initial plan including correct/incorrect, complete/incomplete plans; the terms state* and plan have equal meaning
goal state*	any plan that is correct and complete (and possible other constraints such as for e.g. not more than a maximum number of nodes(i.e. units))

open states or units. An open state is any state of a waste stream with no succeeding unit. An open unit is any unit which has an open inlet port (i.e. no predecessor state).
Unlike to other planning problems such as route planning is the path between initial state* and any goal state* in the context of flow sheet generation identical with the respective goal state* itself. This is because any state* (except the initial state) depicts the alternating sequence of predecessor states* and actions that leads to this state*.
In table 4.13 important terms related to generation of flow sheets are summarized.
The above introduced planning problem can be abstracted by a so called state space search. According to Luger (2001) a state space is defined by the tuple [N,A,S,GD]. Thereby N is a set of nodes in a graph (i.e. states), A is a set of edges (state transitions), S is a subset of N defining the initial state of the problem, GD is a subset of N defining the goal states. Elements of GD are defined by the properties of those elements of GD or the path that leads to these elements. The problem solving is defined by the paths that connect the initial state with the states of GD. Hence state space search defines the means to identify the path between initial state and goal states.

As introduced above a state space search is usefully represented by a graph. This graph can be classified as a tree since there exists a single root node (i.e. initial state), a defined predecessor-successor relationship of nodes (i.e. directed graph) and the absence of cycles. As shown in figure 4.18 (right box) nodes do represent states* (s^*) and edges do represent state transitions (a).

Whereas the state space is represented by a tree does a single state* takes the form of a directed cyclic graph. Thereby do nodes represent model units and edges represent states of waste streams (left boxes in figure 4.18). It is a directed cyclic graph because a single state* may contain multiple root nodes (sources), loops and a clear definition of predecessor-successor relationships between nodes.

In case of forward searching, starting from the initial state, exploring the search space will result in a search tree. The search tree contains all paths connecting initial state with found goal states*. The size of the search tree (b^d) is defined by the branching factor b and by the depth of the search tree d. The branching factor is equal to the number of diversions at each node (see example 3).

4.8.2 Implementing state space search

Algorithms applied for state space search can be described according to the direction of search as well as by their property of being a fully ordered search or partially ordered search algorithm.

For the objective of flow sheet generation a forward and fully ordered search has been applied. Forward chaining (or data oriented search) has been chosen

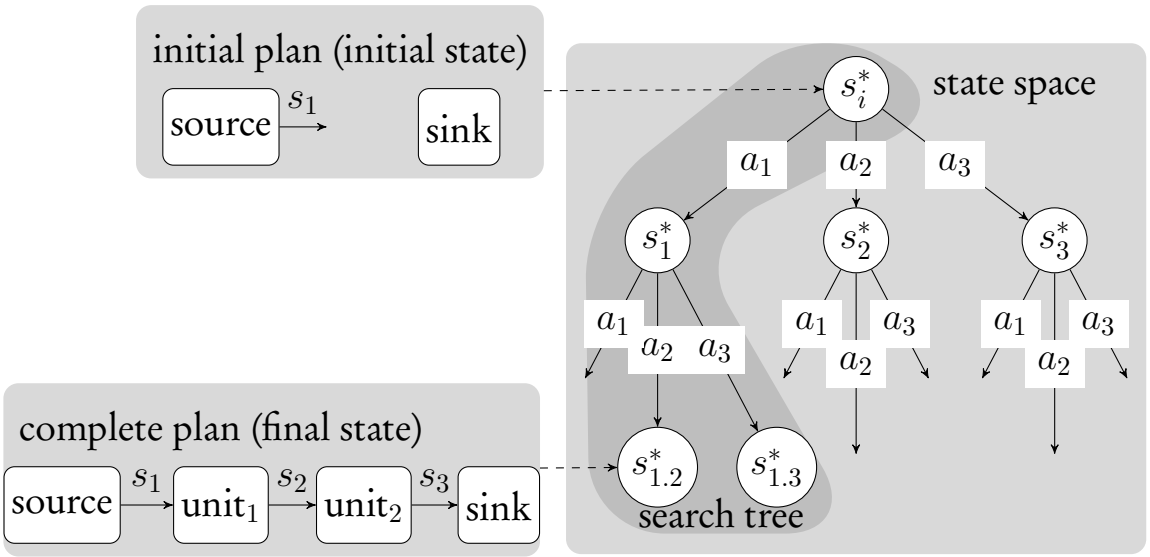


Figure 4.18: Defining state space search and resulting search tree in the context of flow sheet generation (see example 3)

Example 3 - state space
In the example of figure 4.18 there are 3 possible actions (a_1, a_2, a_3) to be applied. Thereby a single action adds a single unit to a state* (e.g. $a_1 \rightarrow$ add unit$_1$). By a preset search depth of 2 and fulfillment of all preconditions the total number of states in the search tree results to $b^d = 9$.

because there are multiple goal states* with an incomplete formulation. In turn, there is a clear formulation of the single initial state to start searching from. Furthermore does the search allows only for a fully instantiated (linear) search since every state* is a unique function of the alternating sequence all predecessor states* and applied actions.
As stated in the introduction the overall objective of a search algorithm is the identification of final states* (i.e. the path to these final states*) in the state space. The difficulty in defining such an algorithms lies in the incomplete definition of final states*. In particular the number of final states* within a given planning task may vary between zero and infinity. Furthermore there exist no clear functional relationship between an intermediate state* and a final state* that can be applied as heuristic function to guide the search. In this work state space size has been limited by the definition of a maximum limit of units within a correct plan. Therefore any state* that violates this criteria is excluded from the state space. Still the identification of an optimal search algorithm is about finding the balance between these two extremes:

- to explore the complete state space and being certain to find all existing final states* to the cost of excessive time consumption and storage demand (i.e. blind search), and
- to explore only part of the search space (through valuation of states* and/or actions) within less time and storage demand to the cost of uncertainty to find all final states* (i.e. informed search, Means-Ends Analysis (MEA)).

4.8.3 Uninformed search

Blind (uninformed) search algorithms are eventually not practicable for the task to identify treatment trains due to the combinatorial explosion of possibilities leading to excessive computation times and required storage resources. However the fundamental set up of a blind search is similar to advanced algorithms such as MEA. Major constraints in the development of a search algorithm in this work has been the implementation of an efficient storage of

the state space while performing the search and the implementation of parallel processing to maximize time efficiency. Since a single state* takes the form of a directed cyclic graph with multiple root nodes the search algorithm has to deal with the formation of duplicate states as well as with the recognition of cycles within a single state*.

In its simplest form a single action within the state space search can be realized by the addition of a new unit to a state* (as shown in figure 4.18) embedded in some control structure to identify completeness and correctness of the respective state*. In this case a single search step must identify at least one open state within the overall state* and connect this state to an open or new unit to generate a successor state*. In this case cycles are recognized and closed automatically through the compliance of secondary preconditions.

However as preparation for implementing MEA, a single search step is described by a two task approach. The functionality of the two tasks are as follows:

Task 1 Define search target

Due to the possibility of multiple root nodes within a state* (i.e. multiple sources) there may exist more than one combination of pairs of open states and open units. The number of combinations is furthermore enlarged by the possibility to introduce merge units. Therefore task 1 accomplishes two sub tasks. First, identify all pairs of combinations between open states and open units. Thereby the number of these combinations is the product of the number of open states and number of open units. Second, identify all possible combinations of open states and the possible addition of merge units. The number of these combinations is equal to $(n^2/2) - (n/2)$, with n being the number of open states within a given state* (see example 4).

Task 2 Execute search target

Within the actual state, transition occurs to generate successor states*. Within this task the following subtasks are processed: check state* in terms of completeness and correctness and add open or new unit to selected open state and generate succeeding state. In case that the insertion of a new unit causes the requirement of a cycle, the start of the cycle has to be identified to close the cycle (see figure 4.20 and example 5).

By the above introduced task distribution it can be assured that no duplicate states* are generated. Within the eventual software implementation of a single search step the two tasks are processed by two independent threads.

Besides the mere functionality of a single search step there is also an efficient storage management of intermediate states* is required. Within this work a double nested array has been used to store intermediate states (i.e. incomplete states). The outer index of this array amounts to the maximum number of units allowed in a plan. The inner index amounts to the stored plans of equal size. Thereby a single plan is stored by an adjacency list.

Through this data structure a search can be processed as depth-first and breadth-first search simply by pulling out the last or first incomplete plan from the array. Regarding blind search both approaches will find all complete plans since the complete state space is explored. However the depth-first approach requires less storage capacity and will yield results faster.

As discussed above, the application of an uninformed search is not practicable for the application to identify feasible treatment flow sheets. This is because of the cardinality of the resulting state space. In the context of the given objective to generate flow sheets there are two strategies that can be applied to control the search. These strategies are informed search and goal driven search.

By an informed search, nodes are evaluated within a search tree. Consequently only nodes are expanded which are closer to the final goal. By a goal driven

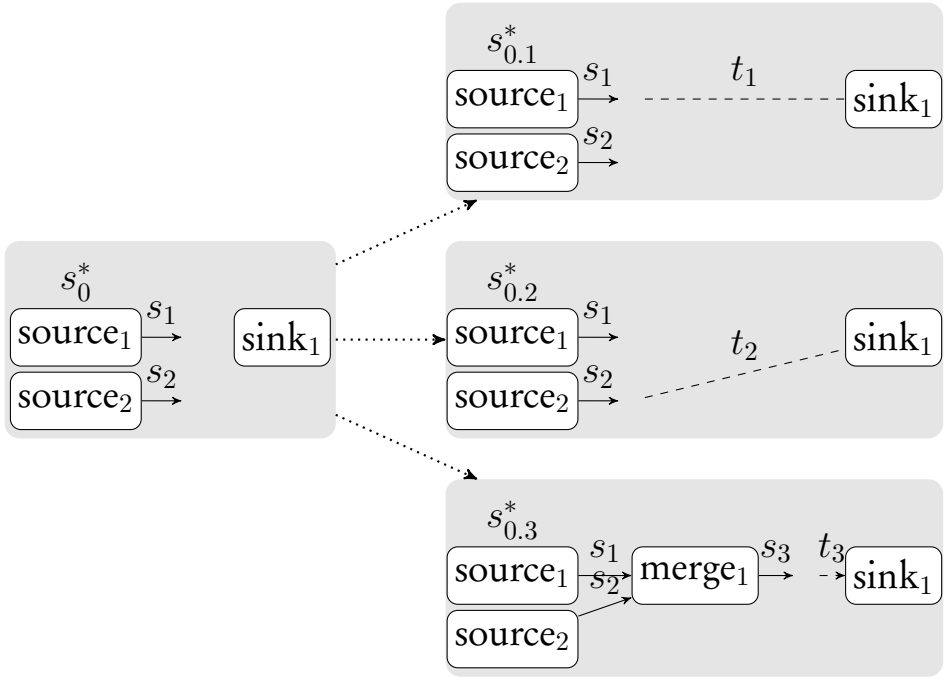


Figure 4.19: Example for the execution of task 1 - define search target (see example 4)

Example 4 - define search targets
Figure 4.19 shows an exemplary initial state* (s_0^*). This state has two open states (s_1, s_2) and one open unit (sink$_1$). Applying task 1 generates two search targets (t_1, t_2) out of the combination of open states and open units. Furthermore there exists one combination ($s_{0.3}^*$) of introducing a merge unit to the initial state, resulting in search target t_3. The resulting three targets are then processed by task 2 of the overall search step.

search only edges are applied that promise to generate successor nodes which are closer to the final goal. In the context of this work evaluation of nodes corresponds to evaluation of states* and evaluation of edges corresponds to evaluating actions to be applied.
An informed search applies a heuristic function h(n) to estimate the cost for the cheapest path for a node n to the final goal g. In other words h(n) estimates the distance between n and g. In route planning h(n) would be the beeline[6] between n and g. For an optimal informed search, h(n) is required to be consistent. A consistent heuristic function fulfills the request of $h(n) \leq h(n') + c(n, a, n')$. This triangle equation states that the sum of heuristic cost of n' and the true cost (c) from n to n' (using action a) is larger

[6]the linear distance

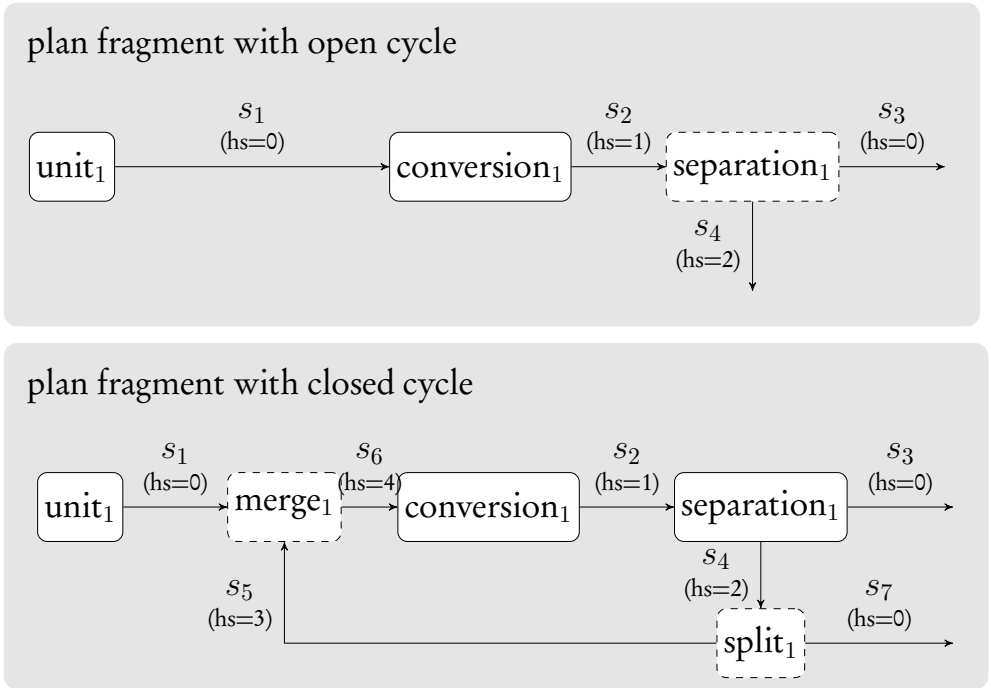


Figure 4.20: Example on closing cycles - position of merge unit is identified through hydraulic status of states (see example 5)

Example 5 - handling cycles
Figure 4.20 (top) shows a fragment of a plan wherein a separation unit has been added (task 2). As a result there are two new states added to the overall state* (s_3, s_4). As soon there is a state with a hydraulic status (hs) that equals 2 a cycle is detected. To close the cycle a merge and a split unit is inserted to the state*. The merge unit is placed at the position behind the last state with hs equals 0 and the split unit is linked to the resulting state with hs equals 2. As a result, the state* holds only states with hs equals 0. On these states (s_3, s_7) new units can be attached.

or equal to the heuristic cost of n. In the context of flow sheet generation the definition of a heuristic function has been discarded. The reasons for this are twofold. The first is again due to the incomplete definition of the final goal and that there are possibly multiple final states*. This makes a clear estimation of some distance measure between any state* and the goal state* difficult. The second reason is related to the steady nature of states*.

The second approach, goal driven search, is implemented in this work by a MEA. In MEA goals are identified and actions are selected according to their ability to address these goals. In contrast to a heuristic function, which quantify the distance between two states are goals solely based on qualitative formulations.

4.8.4 Means-Ends Analysis (MEA)

Essential for the application of MEA is the ability to define a goal between an intermediate state* and the final state* and to evaluate actions according to their ability to address this goal. The fundamental procedure in applying a MEA is shown in figure 4.21. In a first step a goal is identified between the pair of an intermediate state (state^0) and the goal state (sink). Note that the goal is not derived for an entire state* but only for one particular state-unit pair. This first goal valid for the value pair [$\text{s}_0 \rightarrow sink_1$] is referred to as primary goal (g_0). In step 2 all available actions are ordered in terms of fulfillment of the specified goal (g_0). Thereby any unit whose postconditions address all or most goal items of g_0 are ranked best. In step 3 the selected best unit (unit_1) is linked to the before state (s_0). It is thereby presumed that state_0 fulfills all preconditions of unit_1. Next a new goal is derived between the state-unit pair [$\text{s}_1 \rightarrow sink_1$]. Any goal derived after the primary goal is referred to as secondary goal. Within a strict MEA any secondary goal must contain less goal

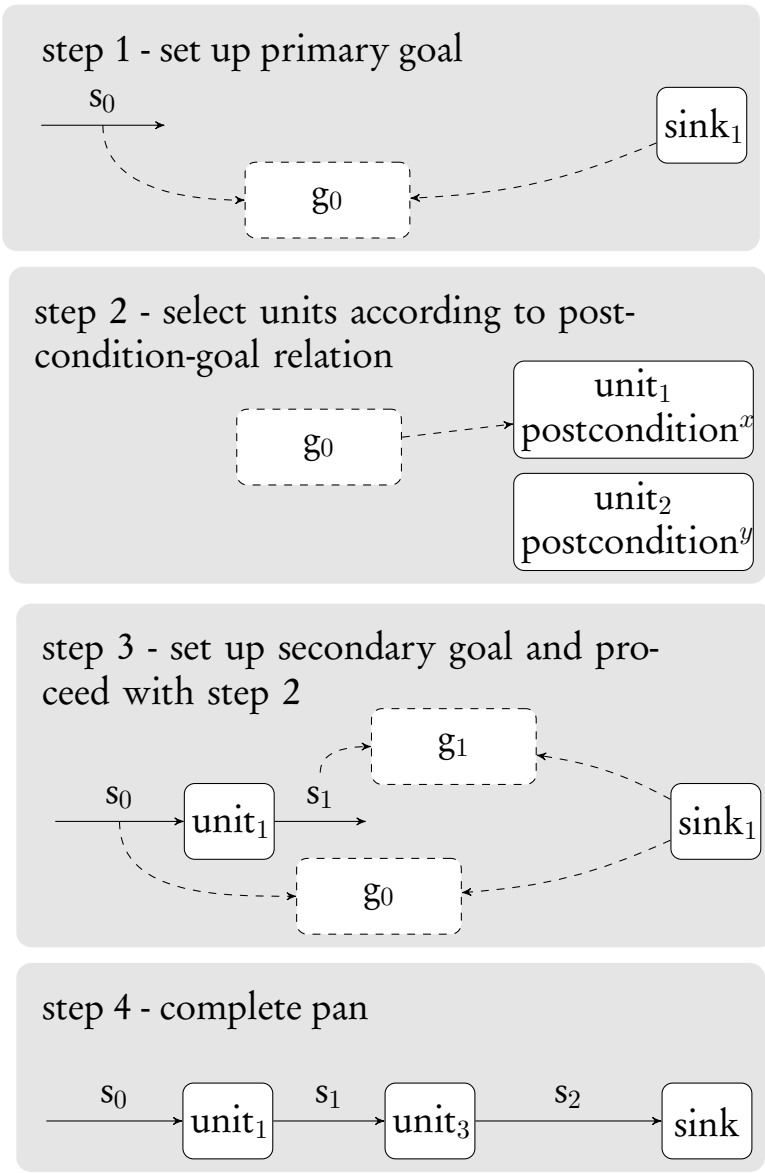


Figure 4.21: Steps in performing a means-ends analysis

items as before identified goals to assure an improvement of successor states toward the preconditions of the aimed sink unit. Step 2 and 3 are repeated until no further goals can be identified. This leads to step 4, which is the identification of a closed path between $state_0$ and $sink_1$. A detailed description on set up of goals is given in example 6.

The above described algorithm applies strict forward searching. However within an advanced MEA backward searching can be applied to used model units whose preconditions are not fulfilled. In figure 4.21 backward search would be required if $state_0$ violates preconditions of $unit_1$. In this case an intermediate goal has to be identified between $state_0$ and $unit_0$. Then units are selected to be inserted after $state_0$ and before $unit_1$.

Table 4.14: Term definition regarding MEA

Term	Definition
goal	qualitatively describes the difference between a state and a precondition of a sink unit. Therefore a goal is a list (or goal set) of goal items.
primary goal	the first goal derived between a state-unit pair
secondary goal	any goal derived after a primary goal has been defined
goal item	parameter-value pair. possible qualitative values are [increases, decreases, changes]
backward search	in case an optimal model unit has been selected according to its postconditions but its preconditions are not fulfilled. Then units have to be identified to be placed before
goal protection	describes the functionality to memorize previous goals to be able to evaluate the improvement of succeeding states

For the implementation of MEA for the objective of flow sheet generation as subject of this work two particularities regarding state* representation have to be dealt with. The first arises of the fact that a single state* may contain multiple root and end nodes (i.e. multiple sources and sinks). The second is related to the representation of a single state as described by the material model.
The first issue has been addressed already with the splitting of a single search step (see section 4.8.3) into two intermediate steps. Within the first intermediate step (see figure 4.19) all possible combinations of open states and open sink units are identified. The MEA is then processed in the second intermediate step.
The second issue, goals are determined based on fractionated quantity sets. Therefore postconditions for each unit are calculated on demand and are not stored in the ontology in contrast to preconditions.

Table 4.15: Exemplary generation of a goal and postconditions from a state (see example 6)

State (s)		Sink unit (su)	Conversion unit (cu)
accumulated		**precondition**	**precondition**
COD/VSS	1,5 -		CS_BOD_soluble $>$ 100 g/m^3
BOD	268,38 g/m^3		
Temperature	15 °C		
VSS	186,66 g/m^3		
$COD_{filtered}$	220 g/m^3		
TSS	186,66 g/m^3	TSS $<$ 100 g/m^3	
$BOD_{suspended}$	161,03 g/m^3		
COD_{total}	500 g/m^3		
$COD_{suspended}$	280 g/m^3		
Q_h	10 m^3/h		$Q_h > 9m^3/h$
$BOD_{filtered}$	107,35 g/m^3	$BOD_{filtered} < 100\ g/m^3$	
fractionated		**goal set (g)**	**postcondition (pc)**
COD/VSS	1,5 -		decrease COD/VSS
Temperature	15 °C		
iBOD5_X_Fraction	161,03 g/m^3		decrease BOD5_X_Fraction
iCODi_X_Fraction	100 g/m^3		
iVSS_unbio	66,66 g/m^3	decrease iVSS_unbio	
iBODu_S_Fraction	120 g/m^3		decrease BODu_S_Fraction
iVSS_bio	120 g/m^3	decrease iVSS_bio	decrease iVSS_bio
iBOD5_S_Fraction	107,35 g/m^3	decrease iBOD5_S_Fraction	decrease iBOD5_S_Fraction
iCODi_S_Fraction	100 g/m^3		
Q_h	10 m^3/h		increase Q_h
iBODu_X_Fraction	180 g/m^3		decrease BODu_X_Fraction

Example 6 - generating goals and postconditions
Table 4.15 shows an exemplary state, sink unit and a conversion unit. The state has a concentration of COD_{total} that amounts to 500 g/m^3 (accumulated state). The sink unit has a precondition on concentration of TSS and $BOD_{filtered}$. The conversion unit represents a reactor for biological carbon removal (see table 4.12) and has a precondition on amount of biodegradable matter and hourly flow. As a result of the intermediate search step 1, the state-unit pair (s-su) is selected for the intermediate search step 2. Within search step 2 for MEA a primary goal is derived (g). Therefore the preconditions of the sink unit are converted into goal items according to the fractionated state. This goal is now used to identify an appropriate model unit. State s is therefore used to generate postconditions on all available model units. As one example the resulting postcondition of the conversion unit (cu) is shown on the bottom right side. The shown postcondition addresses two goal items of the before generated goal. Hence it is estimated that the application of the conversion unit would improve the state towards the fulfillment of the preconditions of the sink unit.

An overview of the implemented search algorithm is shown in figure 4.22. Thereby, the two classes SETUPSEARCH and EXECUTESEARCH represent the two intermediate search steps as introduced on page 157.
The overall search is managed by the class THREADMANAGER. As a first action the initial plan is added to the plan stack - ①. While the search is performed all incomplete plans are stored within the PLANSTACK. The PLANSTACK also eliminates duplicate plans.
The search for complete plans is performed as long the plan stack is not empty - ②. Thereby the THREADMANAGER evokes the first search step (SETUPSEARCH) on each new incomplete plan - ③. In case of more than one open state within an incomplete plan all possible combinations of introduced merge units are identified by recursively executing SETUPSEARCH - ⑤.
For all identified combinations of merge units within an incomplete plan targets are defined between any open state and open sink unit. For each of these incomplete plans with a pair of open state and open sink the second search step (EXECUTESEARCH) is launched - ④.
In the second search step a number of tests are performed on the actual incomplete plan. First, it is checked wether the open state and the open sink can be matched directly (no violation of preconditions of the open sink). In

this case open state and open sink are linked and the plan is returned to the plan stack - ⑦.

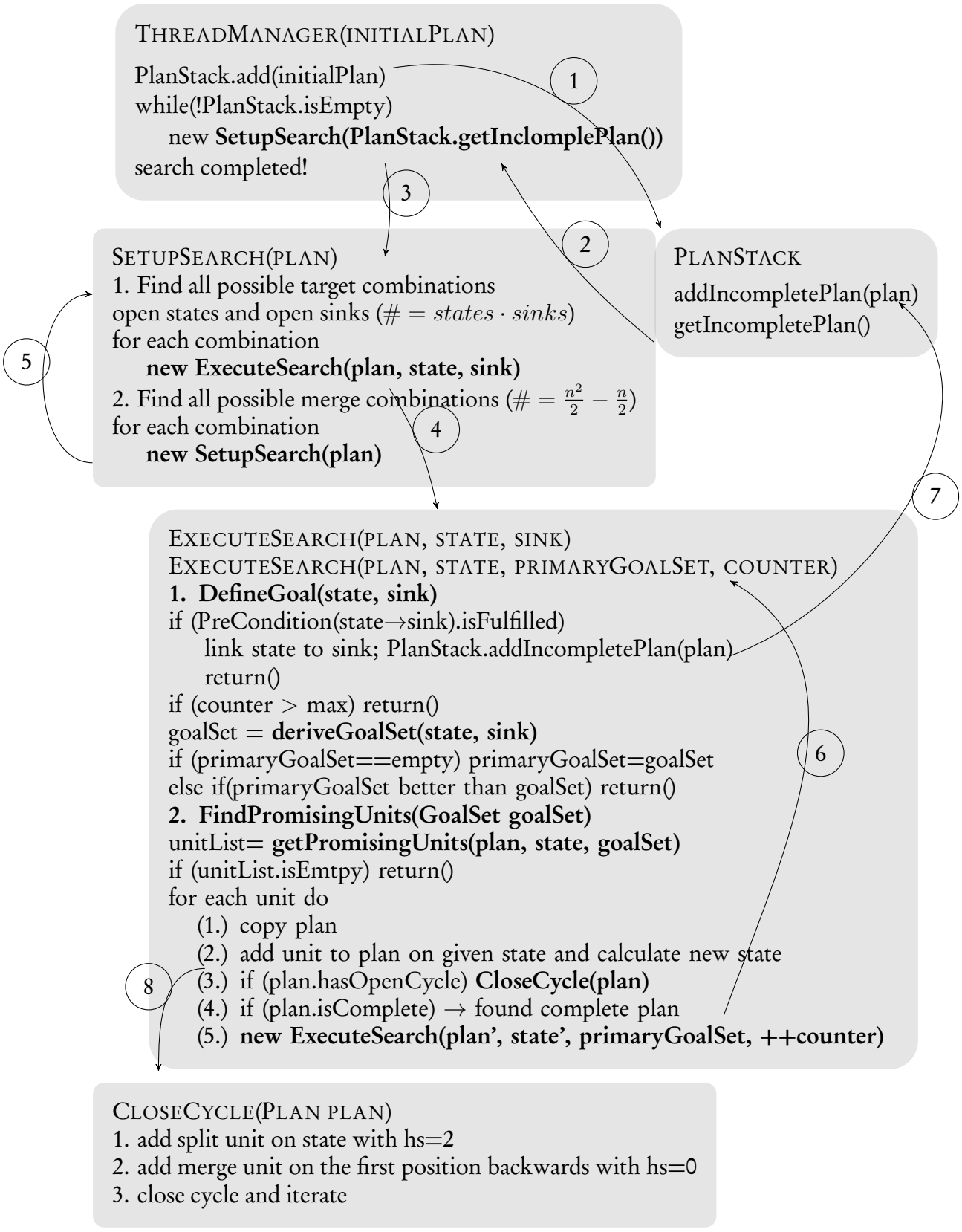


Figure 4.22: Overview on the implementation of MEA

Second, it is checked if the number of cycles exceeds the maximum number of cycles. If a defined primary goal can not reached by the number of maximum number of units per MEA the search is aborted for the actual incomplete plan. Third, a goal is set up based on the open state and open sink (deriveGoalSet). In case MEA is applied for the first time on a particular pair of open state and sink, this goal set is referred to as primary goal. For any succeeding application of MEA on such a pair subgoals are compared to the primary goal to make sure that any further treatment unit brings about an improvement of the wastewater state with regard to the set up primary goal. If any subgoal is worse than the primary goal the MEA is aborted for the actual incomplete plan.
If the actual incomplete plan passes the before tests the actual MEA is applied (Step 2 in EXECUTESEARCH). Based on the derived goal set all units are selected from the knowledge base with have promising postconditions. If no units all can be found that are able to eliminate goal items of the goal set, MEA is aborted. Oppositely for any identified unit a new sibling plan is created wherein the new unit is added to. In case a new added unit demands indicated the existing of a cycle this cycle is closed - ⑧, see also figure 4.20.
In case the resulting plan has no open states or sinks left and furthermore no preconditions are violated a new complete plan has been identified. If not the second search step is recursively applied - ⑥.
A major design criteria of the above described algorithm was to clearly separate single search tasks to be able to be able to process the overall algorithm by multithreading. By this means the classes of SETUPSEARCH and EXECUTESEARCH have been implemented as individual threads. As a result both search task can be processed in parallel by multiple threads to increase the overall search speed.

4.8.5 Controlling search

As discussed in section 4.8.1 the identification of complete plans (i.e. feasible flowsheets) requires the exploration of the state space either by an uninformed/informed search or by a goal driven strategy such as MEA. As discussed before a pure uninformed search can not be applied to the exponentially increasing state space as function of combinations of units (i.e. combinatorial explosion). In turn an informed search can not be defined due to the lack of a property that describes the evolution of plans toward a complete plan as a steady function (i.e. heuristic function). As consequence MEA has been applied to define a goal based search. Through this approach the eventually explored search tree can be minimized in contrast to a blind search which requires

the total exploration of state space. However even through the application of MEA the number of steps to reach possible final states* is still very large[7]. Thereby the number increases with number of incorporated units and by the number of sources and sinks within a given initial state*. By the following criteria the size of state space can be decreased.

Maximum number of units per plan

If this parameter is not specified there is an infinite number of possible complete plans. This is because each unit can be used more than once. Furthermore this parameter must be specified for any search strategy, regardless if uninformed/informed search or goal driven search is applied. The problem lies in defining an appropriate value for this parameter. If the value is chosen to low possibly no complete plans can be identified. In contrast a large value will lead to a very large state space. In the context of this work appropriate values lie between 10 and 20 units. For example if the value is set to 10 in case of use case #1 with one source and two sinks, seven extra units can be added to a plan.

Maximum number of equal units

This parameter defines how often a particular unit can be applied within a plan. In the context of this work values of 2 or 3 haven been applied. In the implemented search algorithm merge units have been excluded by this restriction. It may also be useful to allow for a unit wise definition of this parameter.
The two parameter above are generic parameter that hold for any type of search algorithm. The once following are specific for MEA.

Maximum number of units applied by MEA

The advantage of MEA is that it can rank possible units to add to a plan based on preconditions whereas in a plan search all possible units are added to a plan. This parameter now defines how many units that have promising postconditions eventually are added to a plan. In a very restricted search the value of this parameter would be 1 allowing only the very best options be added to a plan. In the context of this work values between 2 and 5 have been applied.

Maximum number of cycles of MEA

As shown in figure 4.21 the algorithm of MEA adds subsequently promising units to an open state until the primary goal is reached (if guaranteed that subgoals are not worse than the primary goal). This parameter now defines

[7]The number of processed steps for the examined use cases is shown in table 4.19.

how many units can be added to reach the primary goal. In the context of this work values for this parameter have been set to 2 or 3.

Besides these four parameters it is also possible to use the unit wise cost information to restrict the search. At first hand it is not possible to use the total cost of a plan as heuristic function. This is because the optimal total cost of a complete plan is unknown before hand. However as soon the search reveals one complete plan the total cost of this plan can be used as threshold for the ongoing search. Than the search on any incomplete plan that exceeds the threshold will be aborted. Which in turn helps to focus on incomplete plans that may reveal complete plans below the set total cost threshold. However this strategy can only be applied if all cost specifications have a positive sign. This eventually leads to a steady increase of total cost of a plan for any further added unit[8]. In contrast it is also possible to define the cost of a unit with negative sign. For example to use a sink as a water reuse option may have such a negative cost specification[9]. In this case the evaluation of total cost is not steady anymore and the use of a cost threshold would prevent the identification of optimal plans. In the context of this work a cost threshold has been used since sinks have not been linked to cost specifications.
The above described parameter and search strategies limit the number of states within state space, however the efficiency of the search is also affected by the direction of search. As introduced in section 4.8.1 a search can be conducted as a breadth first or a depth first search. In the first, all branches of the search tree are expanded subsequently whereas in the second approach a single branch is expanded as far this is possible before a next branch is expanded. This difference has a severe influence on required storage capacity during a search is performed. In particular breadth first search requires much more storage capacity than depth first. As consequence depth first search shows usually a higher performance. However the advantage of breath first search lies in the immediate identification of duplicate states whereas in depth first search duplicate plans can only be identified as complete plans. It follows that in depth first search more plans are eventually are explored. The appearance of duplicate plans is due to the fact that plans may contain cycles as well by the fact that the search is conducted by multiple threads.

[8]In this way total cost functions as a heuristic function for plan evaluation. Thereby total cost must steadily increase with increasing plan size.

[9]Besides water reuse other features for material recovery or energy production (e.g. anaerobic digestion for biogas production) may be implemented that would justify the use of negative cost (i.e. profit)

A last consequence from the need to reduce the state space is to make sure to define preconditions of units as strict as possible and thereby reduce the possible branching factor during search.

4.9 Application

4.9.1 Introduction

In the preceding sections the developed model as well as the developed search algorithm have been presented. This section now shows some exemplary applications based on the use cases as described in section 3.3.

The use cases have been processed by the developed software for flowsheet generation (see appendix C). Table 4.16 shows the fractionation and accumulation of states of sources of use cases from table 3.3. Fractionation is achieved by the fractionation rules specified in table 4.8.

Table 4.16: Fractionated and accumulated states of sources from table 3.3, only concentrations (g/m^3) and ratios

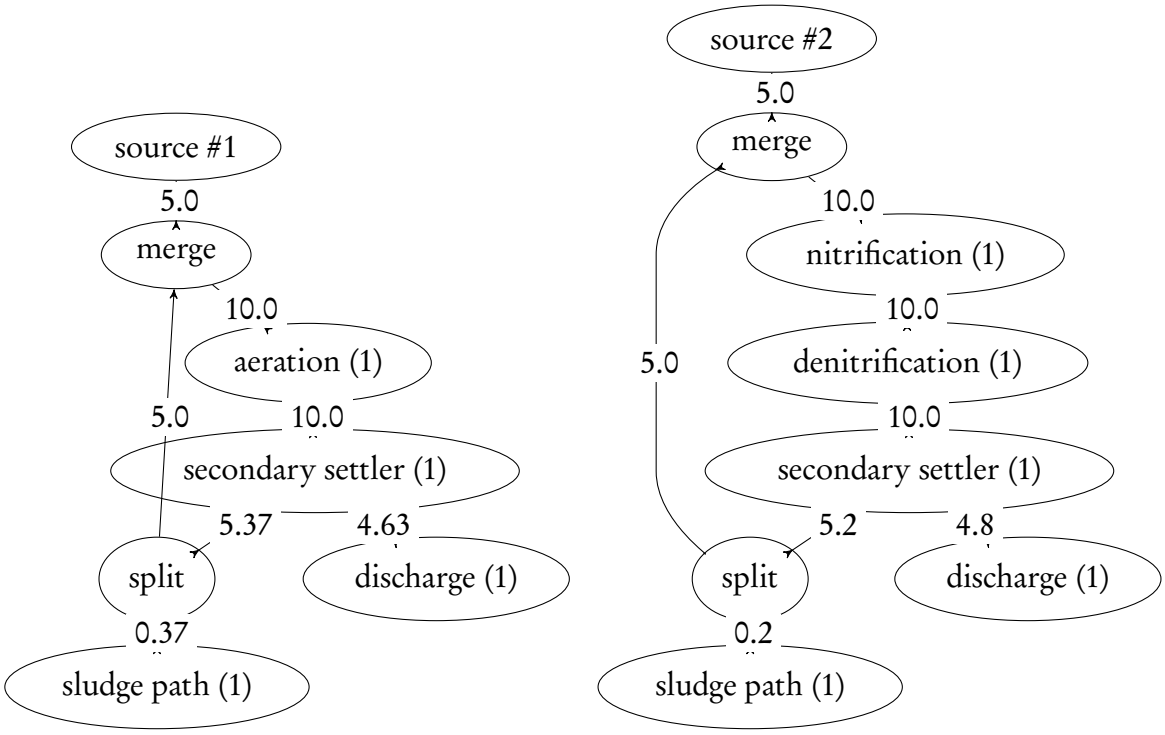
	Fractionated state			
fractions	source #1	source #2	source #3	source #4
iCODiX	100.00	100.00	400.00	200.00
iCODiS	100.00	100.00	400.00	200.00
iBODuX	180.00	180.00	1006.00	24.66
iBODuS	120.00	120.00	670.69	223.56
iBODS	107.35	107.35	600.00	200.00
iBODX	161.03	161.03	900.00	300.00
iVSSbio	120.00	120.00	670.69	223.56
iVSSunbio	66.67	66.67	266.67	133.33
COD/VSS	1.50	1.50	1.50	1.50
NH4*		70.00		50.00
	Accumulated state			
parameter	source #1	source #2	source #3	source #4
COD	500.00	500.00	2470.00	1000.00
BOD	268.38	268.38	1500.00	500.00
$COD_{filtered}$	220.00	220.00	1071.00	440.00
$BOD_{filtered}$	107.35	107.35	600.00	200.00
$BOD_{suspended}$	161.03	161.03	900.00	300.00
$COD_{suspended}$	280.00	280.00	1406.00	560.00
TSS	186.67	186.67	937.36	356.89
VSS	186.67	186.67	937.36	356.89
COD/VSS	1.50	1.50	1.50	1.50
N_{total}		70.00		50.00
TKN		70.00		50.00
NH_4		70.00		50.00
N_{anorg}		70.00		50.00

*Note that no N_{org} is assumed according to table 4.11 if only NH_4 is defined

The result of the software for flowsheet generation is a simple list of feasible plans ranked with regard to total cost. In the context of this section only a few of all identified plans are shown regardless of estimated cost.
Due to the standard fractionation rule set there is a high amount of soluble inert COD which will be passed through the treatment cycle and in consequence will lead to relative high values of total COD to be discharged.
The results shown in this section are supposed to present the technical possibilities of the developed model and algorithms. In contrast, a serious application of flowsheet generation requires much more accurate model unit definitions. This work lies beyond the objective of this work.

4.9.2 Use case #1 and #2

Complete plans with the least number of necessary units for use case #1 and #2 are shown in figure 4.23(a) and figure 4.23(b). Depending on the search parameter set (see section 4.8.5) other complete plans can be identified for the two use cases. The intermediate states of the included model units are shown in table 4.17 and 4.18. Furthermore the evolution of the states during iteration of the cycle flow in each plan is shown in figure 4.24 and 4.25.



(a) Simple cycle for carbon removal - use case #1 (b) Simple cycle for nitrogen removal - use case #2

Figure 4.23: Minimal configuration of flowsheets for use case #1 and #2. Values on edges represent the flow rate in m^3/h

Table 4.17: Use case #1 - outflow concentrations in g/m^3 and flow in m^3/h

Parameter	Source #1	Aeration	Effluent	Secondary settler	Sludge path
Qr	-	5.0	-	5.0	-
COD filtered	220.0	102.67	102.67	102.67	102.67
Temperature	15.0	15.0	15.0	15.0	15.0
Qw	-	0.37	-	0.37	-
COD total	500.0	990.59	111.55	1748.4	1748.4
COD/VSS	1.5	1.35	1.35	1.35	1.35
Qh	5.0	10.0	4.63	5.37	0.37
TSS	186.67	655.73	6.55	1215.38	1215.38
BOD	268.38	16.18	2.53	27.94	27.94
BOD suspended	161.03	13.79	0.14	25.55	25.55
BOD filtered	107.35	2.39	2.39	2.39	2.39
VSS	186.67	655.73	6.55	1215.38	1215.38
COD suspended	280.0	887.91	8.87	1645.72	1645.72

The forming of cycles works as discussed in sections 4.7.4 and figure 4.20. In the present implementation there may be more than one conversion unit enclosed within a cycle. In contrast it is not possible to form cycle in cycle config-

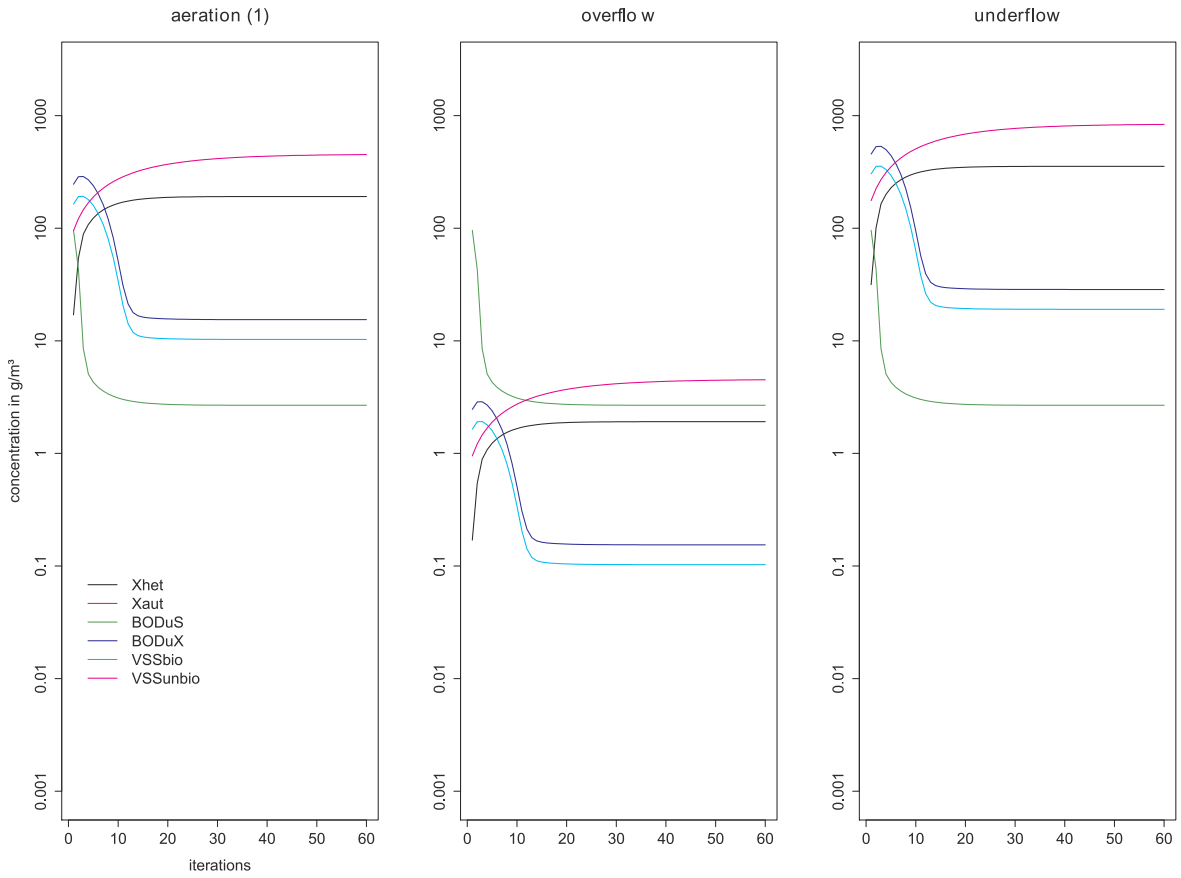


Figure 4.24: Use case #1 - evolution of concentrations on outlet ports

Table 4.18: Outflow concentrations in g/m^3 and m^3/h

Parameter	Source #2	Nitrification	Denitrification	Discharge	Secondary settler	Sludge path
Qr	-	5.0	5.0	-	5.0	-
COD filtered	220.0	116.14	103.5	103.5	103.5	103.5
Temperature	15.0	15.0	15.0	15.0	15.0	15.0
NH4	70.0	0.49	1.71	1.71	1.71	1.71
TKN	70.0	0.49	1.71	1.71	1.71	1.71
Nitrate	-	46.9	35.16	35.16	35.16	35.16
N total filtered	70.0	47.39	36.87	36.87	36.87	36.87
anorganic Nitrogen	70.0	47.39	36.87	36.87	36.87	36.87
Qw	-	0.2	0.2	-	0.2	-
COD total	500.0	2061.29	2020.0	122.58	3771.39	3771.39
COD/VSS	1.5	1.25	1.25	1.25	1.25	1.25
Qh	5.0	10.0	10.0	4.8	5.2	0.2
TSS	186.67	1556.04	1530.31	15.25	2928.78	2928.78
BOD	268.38	31.96	12.83	3.23	21.69	21.69
BOD suspended	161.03	17.52	9.7	0.1	18.56	18.56
BOD filtered	107.35	14.44	3.13	3.13	3.13	3.13
N total	70.0	47.39	36.87	36.87	36.87	36.87
VSS	186.67	1556.04	1530.31	15.25	2928.78	2928.78
COD suspended	280.0	1945.15	1916.5	19.08	3667.89	3667.89

urations as it could be required for nitrification-denitrification processes. The root of this limitation lies in the strict forward chaining of plans that comes to the prize of being myopic. This limitation can be solved by an improved algorithm for cycle configuration.

Looking on the evolution of parameter VSS and particulate BOD for aeration in figure 4.24 shows the correlation between parameter that resemble oxygen equivalents and parameter that resemble matter equivalents as described in section 4.5.3. In the succeeding settling unit particulate matter is separated from the overflow, which as consequence leads to a reduction of biodegradable particulate matter which is linked to VSS.

In the example of use case #2 it can be seen in figure 4.25 that the iteration cycle takes much longer to come to a steady state. At present a single iteration step is equal to the HRT of the reactor of the respective conversion unit. Eventually the iteration is stopped if all states before and after calculation do not differ on each quantity by the value of 0.1. However if an iteration takes more than a maximum number of iterations the respective incomplete plan is discarded. The preset value of this maximum number of cycles is set to 100.

It can be seen for use case #1 and #2 that the general identification of flowsheets works as desired and the change of wastewater states is calculated according to

linked models. However the two examples show also a conceptual fault. Both sources #1 and #2 have the same amount of COD, only source #2 has additionally ammonia included. Consequently the flowsheet shown in figure 4.23(a) would be incorrect. Through the lack of a nitrogen source no biological treatment under aerated conditions could be applied for carbon removal. In both flowsheets the aeration unit and the nitrification unit share the same processes except the fact that the process of growth of heterotrophic biomass in the aeration unit has no stoichiometric term for ammonia. The design conflict behind this issue can be summarized by the following two perspectives.

first perspective

It is assumed that a wastewater state is fully characterized by a given set of parameter. It follows therefrom that if a parameter is not included in the related state set this parameter (i.e. substance) is not contained in the respective wastewater stream. This assumption relates to CWA. In the context of the above given example of figure 4.23(a) the search should not reveal any feasible plan because only biological carbon removal (under aerated conditions) is able

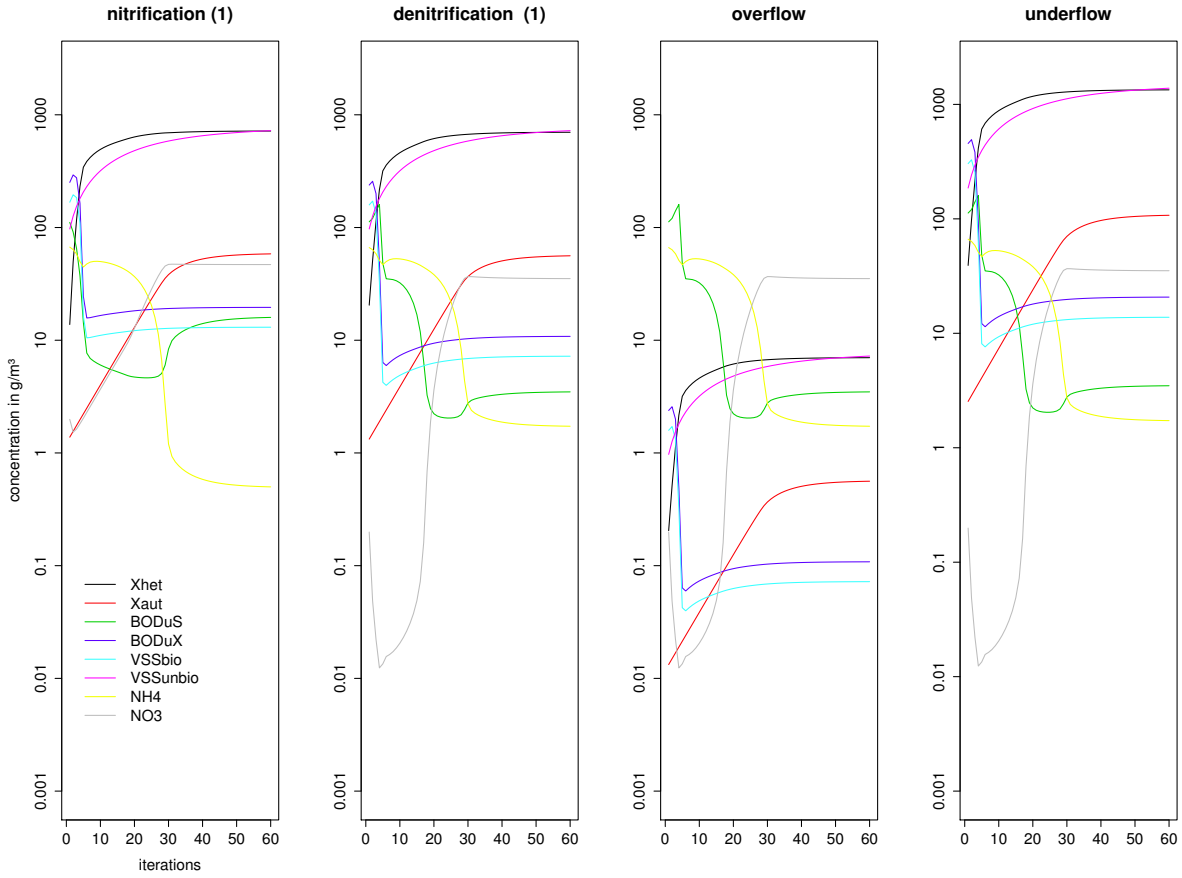


Figure 4.25: Use case #2 - evolution of concentrations on outlet ports

to purify the given COD contamination sufficiently out of the units specified in table 4.11 and 4.12. Another possibility would be the definition of an extra unit that represents additional dosage of ammonia. Such a unit can than be added before the actual aeration unit to enable growth of heterotroph biomass for carbon removal. The difficulty with this assumption is that it can not be guaranteed that all substances within a given wastewater are covered by quality parameter. Nevertheless it must be guaranteed that all parameter that are required for modeling are included in the given parameter set. Consequently this approach suggests that a particular treatment technology is represented by a single modeling approach which in turn requires a minimum parameter set.

second perspective

It is assumed that a wastewater state is not fully characterized by a given set of parameter. It follows therefrom that if a parameter is not included in the related state set the parameter value (i.e. substance concentration) is unknown. This assumption relates to OWA. This assumption leads to the strategy to choose the modeling approach for a particular treatment technology based on the given parameter set. Than simplified modeling approaches are applied when a scarce parameter set is provided and more complicated modeling approaches are used if an extensive parameter set is provided. Hence this assumption would require the representation of multiple modeling approaches with varying model granularity. However the hope related to this assumption that a more detailed wastewater specification would lead to more profound flowsheets in comparison to a less detailed wastewater specification is misleading. Because an incomplete wastewater specification may in its worst case lead to wrong flowsheets.

Evaluating the two possibilities described above the first approach should be applied for future developments of the overall approach. To overcome the problem of insufficient wastewater characterization a library of a branch or even production process wise description of expected wastewater characterization should be implemented. The required input information would than cover the specification of measured quality parameter and the additional specification of the production process or industrial branch related to the wastewater source. Required but not specified parameter can than be supplemented from a library.

4.9.3 Use case #3

Figure 4.26 shows three flowsheets derived from use case #3. In particular figure 4.26(a) shows the configuration with the least number of required units. Due to the large amount of COD the application of two fermentation units is required before conventional aerobic biological treatment to meet the preset effluent standards.
The flowsheets shown in figure 4.26(b) and 4.26(c) are similar to the one shown in figure 4.26(a) except that they additionally include two separation und merge units. Obviously the two flowsheets make no sense from a practical perspective. However, according to the set constraints on the search algorithm these two flowsheets are correct and complete plans. The search algorithm at present already includes tests to filter out faulty patterns such as settlers whose under- and overflow are directly rejoined in a succeeding merge unit.
The problem in defining such filter rules is to find the balance between very strict rules to filter out as much inconsequential plans on one side and to make sure that no correct plans are discarded during search.

4.9.4 Use case #4

Use case #4 is similar to a brewery wastewater as shown in table 3.1. As indicated in figure 3.4 a possible solution would consist of a combined fermentation and a conventional nitrification and denitrification cycle. As result of the search figure 4.27 shows three results of use case #4. In all three examples two sequential cycles have been used. A fermentation unit has only been used in the example shown in figure 4.27(c). There are two reasons for this behavior. First, the conversion units for nitrification and denitrification as specified in table 4.12 have both a fixed reactor volume as well as fixed values for waste flow and recycle ratio. Hence a single nitrification and denitrification cycle is not sufficient to treat the waste stream. Second, search parameter (section 4.8.5) to limit the number of units per cycle is set to the value of 2. Hence the search can only identify plans with two cycles that contain less or equal to two conversion units.

4.9.5 Use case #5

Use case #5 has been set up to test the overall approach with regard to identify plans based on multiple sources and sinks. Within use case #5 there are two sources (#2 and #4) incorporating COD and ammonia with varying concentrations. Three results for use case #5 are shown in figure 4.28. In each of the three examples there is an independent treatment chain for each source. In

Table 4.19: Overview on search results

UC	max # of units per plan	max # of units per cycle	max # of units for MEA	max # of units per type	plans	steps*
1	12	2	5	2	22	2691
2	12	2	2	2	2	2558
2	14	2	5	2	22	10958
3	12	2	3	2	2	3688
5	14	2	3	2	5	104265

*The processing speed in step per second depends on the applied computation power. In this work a speed between 5 to 1 has been experienced.

contrast in all search results there has been no plan where both sources have been merged to treat the combined wastewater. This is due to the absence of a split unit that divides a single waste stream in to two. The split unit as it is implemented now, only splits a waste stream if the hydraulic status is equal to 2, in the case of a secondary setter. If use case #5 would only contain the two sinks of sludge path and discharge the two sources could have been merged an treated together in a single treatment chain. Two options for further development can be concluded therefrom.
First, the search algorithm must allow to use sink units optional. In use case #5 the discharge sink could be excluded as long the treated wastewater can be linked to the sink unit for water reuse. In particular the definition of a complete plan would accept a plan with open units as long there are no open states.
Second, the knowledge base on treatment units must include split units that split a given waste stream by preset ratios (e.g. 1:1). However the implementation of such a split unit into the search algorithm is difficult since it has no meaning full post condition. For the MEA algorithm has no reason to use such a split unit.
especially if a state* contains more than one open state and more than one open sink (e.g. more than one source) the branching factor increases proportional to the number of open states times open sinks.

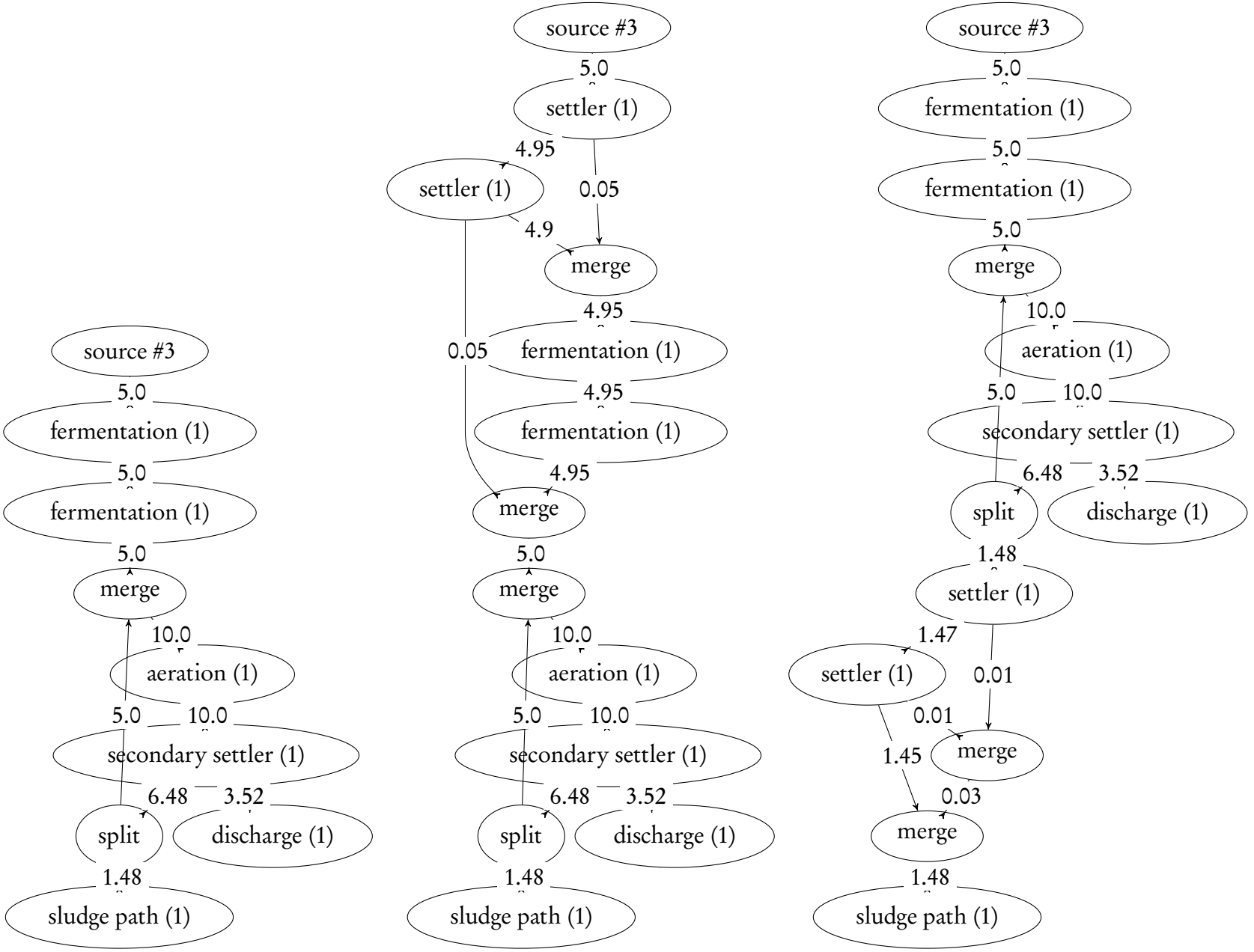


Figure 4.26: Three examplary results of use case #3

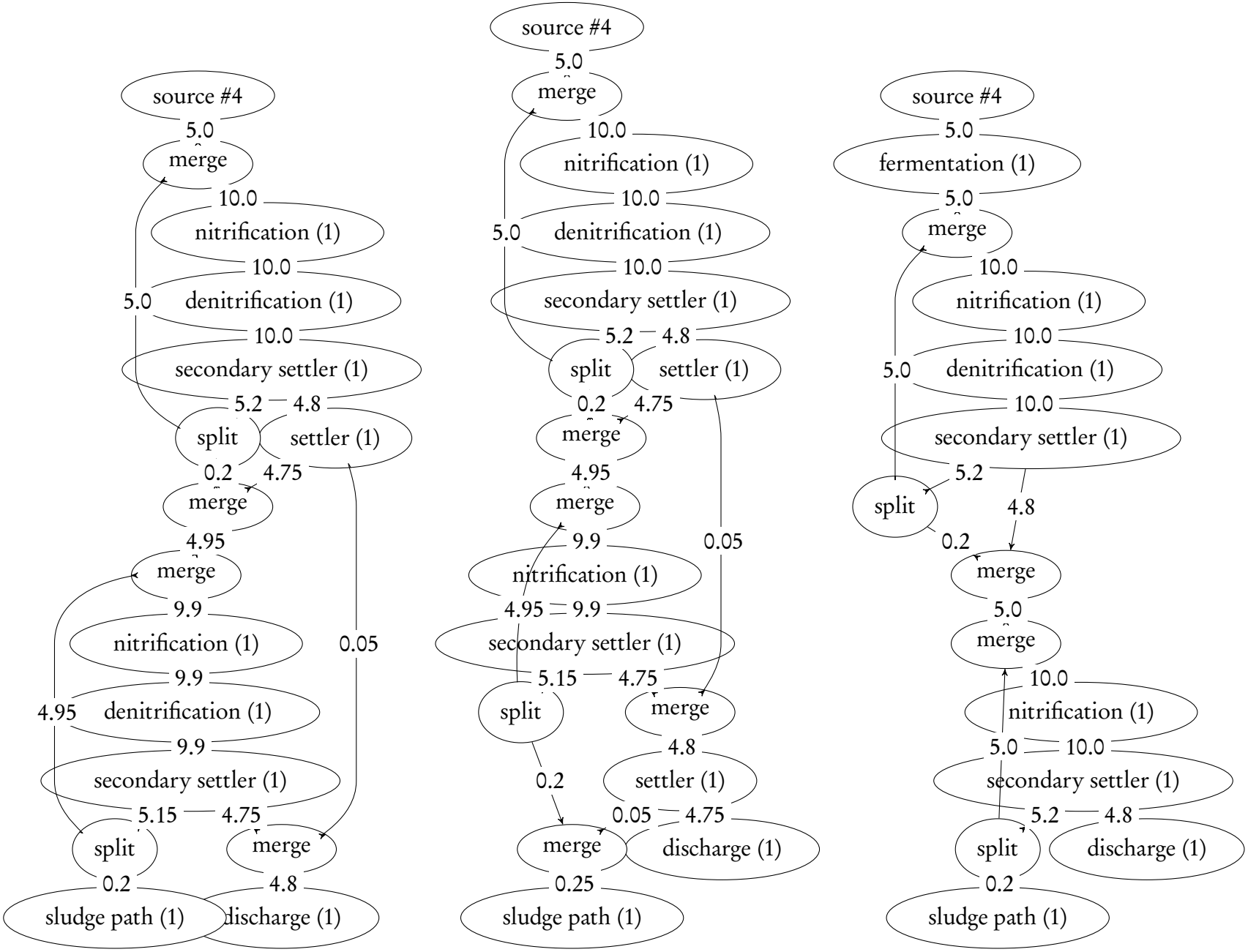


Figure 4.27: Three exemplary results of use case #4

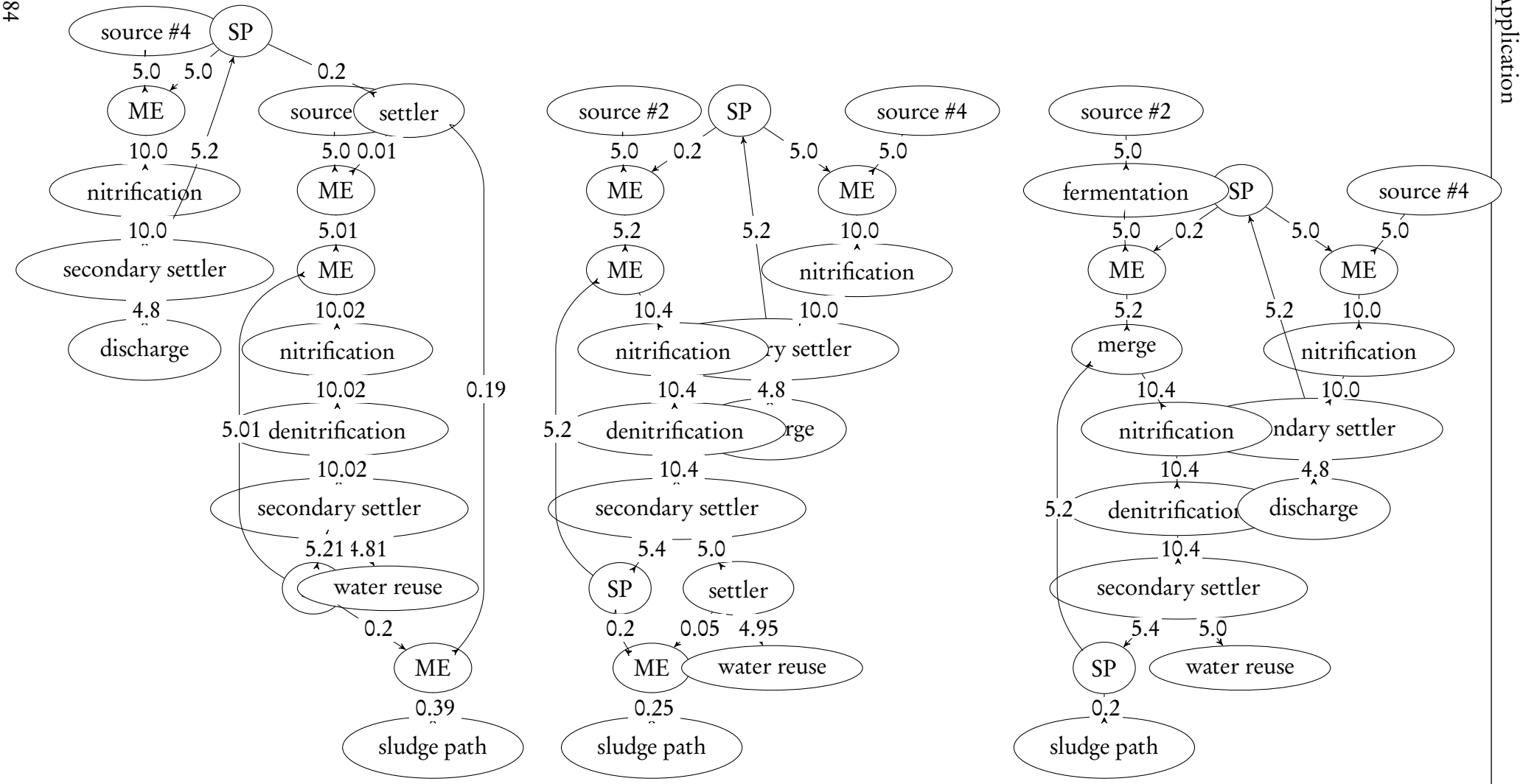


Figure 4.28: Three exemplary results of use case #5 (SP - split unit, ME - merge unit)

4.10 Discussion

In the previous section a number of use cases have been applied to show the functionality of the developed tool for flowsheet generation. It has been shown that the overall approach technically functions as desired. The identified flowsheets are correct with respect to the set constraints through specified preconditions as well as by the limited list of available treatment units. In contrast, to produce a serious practicable application still requires a number of improvements.
This section discusses main advantages and disadvantages of the approach with respect to the key development aspect such as model representation, process model and the applied search algorithm. Furthermore suggestions with regard to the further development of the model as well as to the application are discussed.

Layer model representation

If the purpose is to use a formal declarative language such as OWL(DL) for model representation the set up of a layer model representation equal or similar to the one discussed in section 4.4 is inevitable. Although required modeling aspects have been successfully represented in this work into an ontology it appeared that the implementation process is error prone. Especially the translation of mathematical equations into tree like representation by the `Term` concept is ponderous. Even more the thereby implemented equations can not be validated automatically. So far the declarative model can only be extended or adjusted either by editing the OWL file directly or through the use of an ontology editor such as Protégé as it has been done in this work. Certainly the development of a tailor made GUI for editing the declarative model would ease the addition of new knowledge (e.g. on parameter, treatment technologies) and to manage already represented knowledge. Another possible improvement would be the use of scalar variables. Then each parameter would have not only a value assigned but also a unit. Consequently represented equations could be automatically validated and balanced out to the correct unit. At present the implemented software assumes a particular unit for concentrations and flow. The use of scalar variables in the context of model representation is described in Bogusch and Marquardt (1997).

Process model

The list of model units described within the process model can be referred to as data base of treatment units. Hence this list stores the options of how wastew-

ater streams can be treated. In the present implementation all model units are based on the six generic model units as described in section 4.7.2. In particular only the two types of conversion units and separation units can be used to describe particular technologies for wastewater treatment. A required development would be the introduction of another generic unit type that allows for concurrent separation and conversion processes. Technological examples are UASB reactors or SBR units. Besides units that allow for qualitative state change it would be advisable to introduce units for quantitative state change. The technological equivalence is set by equalization tanks. To eventually be able to implement equalization tanks the further definition of hydraulic parameter would be required that describe the temporal variation of flow.

Search algorithm - MEA

It has been shown that the applicability of the overall approach depends mainly on the efficiency of the applied search algorithm. The key in improving this efficiency lies in the ability to differentiate between promising and irrelevant incomplete states* within state space. MEA itself does not offer this ability to evaluate states* in state space. Instead MEA is used to decreased the branching factor on each state* through evaluation of options (i.e. through postconditions of model units). A reasonable approach in increasing overall search efficiency lies in enhancing MEA. The present implementation of MEA only allows for forward search. This means, that through MEA options for succeeding model units are evaluated under the restrictions that preconditions of these options are fulfilled. But what if a model unit shows the most promising postconditions but its preconditions are not fulfilled by the respective state? In case of only applying a forward search this option must be discarded. In case of additional backward search a subgoal can be defined to identify possible predecessor units to be able to include the respective promising model unit. Through this measure the parameters of MEA (section 4.8.5) can be set to more restrictive values which leads to a reduction of state space size. If the size of state space is approximated by depth, d and branching factor, b according to b^d this would lead to a decrease of b.

Besides improvement of MEA also the means for evaluation of states* can be achieved. As stated in the previous section does the state space contains a large amount of irrelevant states*. The objective must therefore be to recognize and eliminate those plans in early stages of development to avoid unnecessary calculation effort. An example of such a irrelevant pattern is where both out-

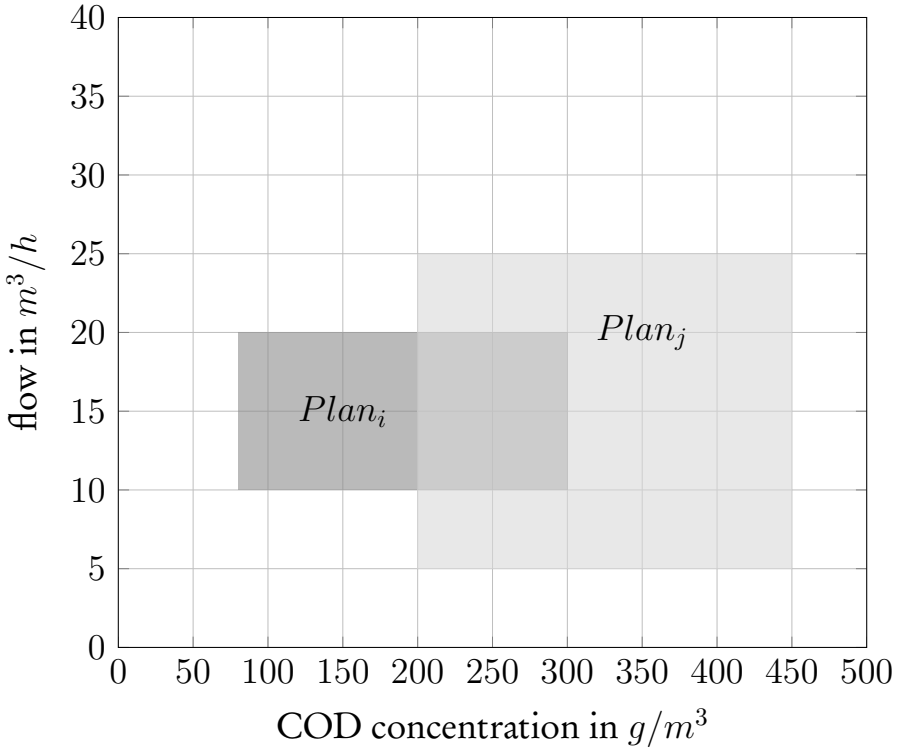
flows of a separation unit are connected to a succeeding merge unit (see also figure 4.26(b) and 4.26(c)).
As indicated already in section 4.9.5 another possibility to enhance the search algorithm lies in the way sink units are processed within the search. In its present implementation given sink units from the initial plan must be included within every identified complete plan. As an alternative the search must be able to omit given sink units from complete plans if this is possible.

Further possibilities for the identification of flowsheets

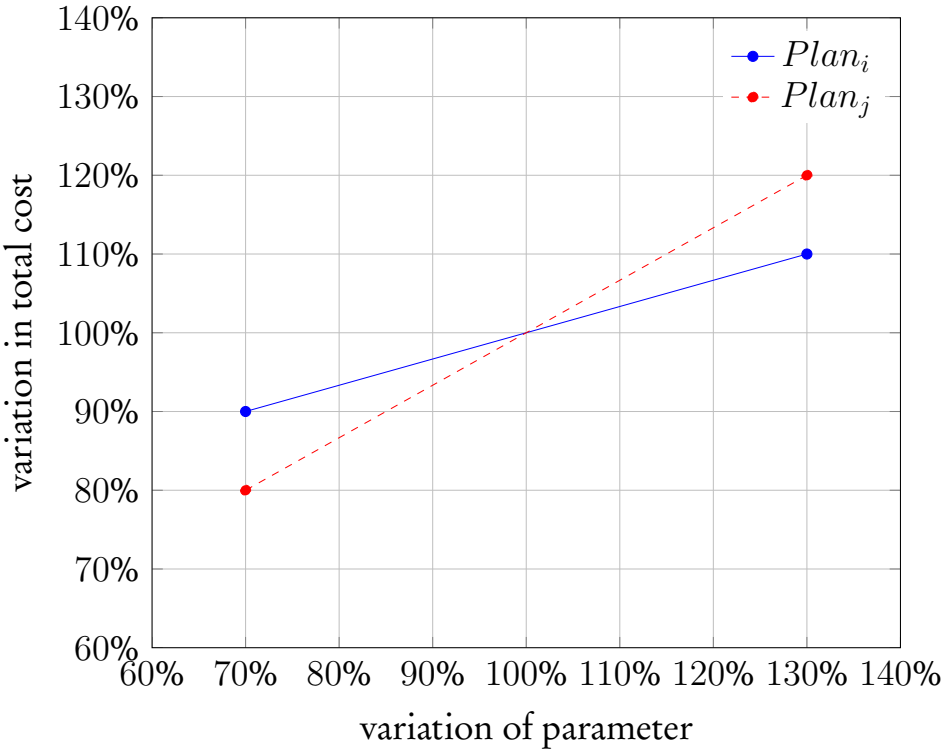
Two functionalities that have not been presented in the application section (4.9) are (i) the ability to add any other model units to an initial plan beyond sources and sinks and (ii) to define negative cost definitions on units. The first functionality opens the possibility to upgrade existing flowsheets by additional treatment options. For example may an existing treatment chain be upgraded to allow for further incorporation of sources or to evaluate possibilities of water reuse. Therefore existing treatment units can be reused within upgraded flowsheets. The second possibility can be used to define negative cost specifications to sources that resemble sources for water reuse. Within the search for complete feasible flowsheets sources with a negative cost information lead to a lower total cost of plans where water reuse is considered.

Evaluation of flowsheets

As discussed in the application sections, the result of the flowsheet finder algorithm is a list of flowsheets ranked with regard to total cost. To increase the value of the result the two following approaches can be applied. The first is the execution of a sensitivity analysis and the second is to blend all identified complete plans in order to identify reoccurring patterns throughout all identified plans.
In particular, sensitivity analysis can be applied to estimate the robustness of plans with regard to input parameter variation as well as to estimate the influence on total cost of plans with regard to input parameter variation. The first case is shown exemplary in figure 4.29(a). If the initial state has a specified value for COD of 250 g/m^3 and a value for hourly flow of 15 m^3/h both plans i and j would be sufficient to treat the respective wastewater. Changing the values of both parameters would reveal the application limits of both plans. In particular the areas in figure 4.29(a) resample the valid application range. If more than two parameter are changed different forms of visualization must be applied. In figure 4.29(b) an exemplary sensitivity analysis on total cost with



(a) Application limits



(b) Influences on total cost

Figure 4.29: Options for evaluating list of flowsheets

regard to input parameter variation is depicted. Thereby the influence on total cost of plans can be analyzed on parameters such as specific energy cost (e.g. Euro per kWh), specific construction cost (e.g. Euro per m^3) or change of input parameter (e.g. COD, flow, etc.). At present any form of sensitivity analysis is not implemented yet and is hence subject for further development.

Fields of application

The intended area of application of the planning tool developed within this work is the design of wastewater treatment flowsheets for small industrial companies. Emphasis has been put on the consideration of multiple sources and sinks. Regarding this, the functionality of the design tool as it is implemented at present allows for the identification of water reuse potential for given treatment objectives. Beyond this the planning tool can be extended in its functionality to be applied in the wide area of resource recovery and design of energy neutral wastewater treatment facilities (Verstraete and Vlaeminck, 2011). Through its generic set up the material model of the planning tool can be extended to those substances relevant for resource recovery. Furthermore alternative process technologies (e.g. anaerobic technologies, SHARON-Anammox process) would have to be implemented into the process model.

Comparison to other WWTP design tools

Comparing the presented planning tool to existing approaches as discussed in section 2.6.4 the following similarities and differences can be identified. First of all the presented approach falls into the category of knowledge-based approaches in contrast to optimization based approaches. In this regard the planning tool circumvents the problems in superstructure generation as it is experienced in optimization based approaches. With regard to knowledge-based approaches the presented planning tool avoids the use of past experiences (i.e. rule based approach, cased base reasoning.) The application of a means-ends analysis has been applied in past planning approaches. However the planing algorithm of this work has been further developed with regard to consideration of multiple sources and sinks as well as to the consideration of cycles within treatment configurations. An essential innovation in contrast to existing planning tools lies in the application of a declarative knowledge base for generic process and material representation. A drawback of the presented algorithm lies in its tendency of being myopic as result of strict forward planning. In contrast optimization based approaches do not show this disadvantage. However

this handicap of the presented planning tool may be cured by a post processing of identified plans to identify optimal solutions.

5
Conclusion

5.1 Summary and research contribution

The demands on treatment of industrial wastewaters do not only involve the achievement of present legal effluent standards but also the identification and application of the most cost efficient treatment technologies to contribute to the overall economy of manufacturing companies. Thereby cost efficiency can be increased through water reuse and by the implementation of treatment trains that consider multiple sources of wastewater from different production steps separately instead of simple end of pipe treatment. The design of an optimal treatment facility usually starts with the identification of different flowsheets (i.e. treatment trains) that contain a combination of different treatment units. Thereby each unit within a treatment train targets on the purification of particular wastewater constituents (e.g. extraction of particulate matter, removal of organic matter etc.). In the run of the planning process the identified flowsheets are compared and evaluated regarding cost efficiency and robustness by engineer experts based on experience and use design tools to eventually identify the optimal flowsheet for the given task of wastewater treatment. The identified flowsheet will then be the basis for the practical implementation of the treatment facility.

This thesis presents a software tool to aid the planning process of industrial treatment facilities. Thereby the software tool is able to autonomously generate feasible flowsheets based on initial flowsheets that contain multiple sources (e.g. wastewater streams of various production sites) and sinks (e.g. discharge point, quality demands for water reuse). Sources are defined by wastewater characteristics through water quality parameter and sinks are defined by limit values to be met.

The list of available treatment units to be included within a flowsheet contains biological conversion units for carbon and nitrogen removal and separation units such as primary and secondary settlers. The algorithm for flowsheet generation can also consider cycles regarding biological treatment processes.

Identified results are presented as a ranked list of feasible flowsheets according to estimated total cost of each flowsheet. The representation of flowsheets is based on unit operations. Thereby the degree of detail is set to the scale of treatment reactors wherein each treatment technology is represented by a single model unit. It follows thereof that a single flowsheet is a set of model units which are connected through wastewater streams.

The scientific contribution of this work is twofold. First, a logic based modeling language has been used to represent required modeling knowledge within a single model framework. This model framework functions as a knowledge base (KB) stored apart from the software tool itself. The major advantages of this approach are: (i) the possibility to independently add or change knowledge apart from the software tool (e.g. add new treatment units or water quality parameters); (ii) to gain the possibility to consider interrelations between different model aspects (e.g. the relation of treatment units and their treatment effect on particular water quality parameters).

The second scientific contribution is the development of a search algorithm which is able to identify feasible combinations of treatment units without the need of additional design rules for flowsheet generation. Commonly the problem of flowsheet design is solved by the use of rule based or case based approaches. In these cases design decisions are derived on past experiences or existing solutions. Through its generic design the approach of this work opens the possibility to identify combinations of treatment units that are beyond standard solutions.

Application of a logic based language for model representation

The innovative feature of this work is the use of the Web Ontology Language, OWL(DL) to define a KB on required modeling aspects. OWL(DL) is a free web standard endorsed by the World Wide Web Consortium (W3C). Its language theory originates solely on description logics in combination with standardized language syntax based on RDF. Hence the use and development of OWL(DL) is free from proprietary claims.

The necessities and benefits that explain the use of an OWL(DL) based KB are the following. The KB is stored independently from the software tool, it can be easily extended and even offers model reuse beyond the original application. The expressiveness of OWL(DL) is richer as compared to conventional database storage techniques. This allows for the representation of heterogenous modeling aspects in a single model framework.

Related to this, treatment units can be represented in a generic way. Commonly the treatment efficiency of treatment technologies are described in literature depending on industrial branch and respective water quality parameters. In this work a material model has been developed to allow for generic formulation of treatment efficiency through the use of conventional mathematical models. The ability to address different industrial branches, wastewater peculiarities can be customized through the definition of branch specific fractionation rules. Through fractionation rules, given data on wastewater quality is interpreted with regard to properties of wastewater constituents and present substances.

An OWL(DL) based KB offers furthermore for automated model checking in terms of its logical consistence through the use of freely available automated reasoners. This supports the development of the KB since logical inconsistencies can be automatically identified.

Another feature offered through use of OWL(DL) is the use of logical consequence to derive implicit knowledge (cf. section 2.4.2.1). In this case facts can be derived from a KB that have not been explicitly defined but follow as logical consequence from facts that have been declared within the KB. In the context of flowsheet generation this has been used to declare relations between water quality parameter.

The strength of OWL(DL) lies in the formal representation of declarative knowledge. Hence the declaration of relations between water quality parameter, substances and properties can easily be represented within OWL(DL). In contrast complex modeling constructs can not directly be represented within OWL(DL). Regarding this work, this involved the representation of conditions, mathematical models or complex model constructs such as treatment units.

To encounter this problem a layered model representation has been developed. Therefore language elements of OWL(DL) have been used to define auxiliary modeling concepts. Auxiliary modeling concepts declare the concept of conditional terms, algebraic terms and processes. Based on these auxiliary concepts mathematical models can be defined as binary trees as well as treatment units as a set of conditions and mathematical models.

Besides knowledge representation a tailor made interpretation structure is required for knowledge processing and application. In the case of the software tool for flowsheet generation a parser has been developed to translate equations

which are represented in OWL(DL) to be able to calculate treatment efficiencies of represented treatment units.

An algorithm for autonomous generation of treatment flowsheets

The heart of the software tool is the algorithm for identification of treatment flowsheets that function without the need of additional expert knowledge regarding flowsheet design.
The challenge in finding feasible flowsheets lies in the effective and determined exploration of the state space of possible treatment unit combinations.
For common applications the number of elements in the state space out-rules the possibility to apply a blind search on the entire state space. This is because the number of combinations is an exponential function of the number of different treatment units and flowsheet size. This phenomena is refered to as combinatorial explosion. A further limitation lies in the requirement that flowsheets can only be identified by a strict sequential forward search, starting from wastewater sources stepwise toward given sinks.
Commonly the problem of combinatorial explosion within flowsheet generation is circumvented by the use of additional design rules based on expert knowledge (e.g. rule based or case based approaches). In this work the use of an additional rule base has been consciously avoided to achieve a generic design approach.
This has been achieved through an adapted means ends analysis (MEA), developed and applied as a goal based search algorithm. In this approach possible treatment units are selected based on identified goals and treatment unit specific post conditions. In this context goals are derived from limit values related to sinks (e.g. nitrate must be reduced). Oppositely post conditions qualitatively state the expected effect of each treatment unit with regard to an inflowing wastewater stream (e.g. this treatment unit reduced nitrate).
The effective use of goals and the selection of treatment units based on post conditions is only made possible through the advanced representation of water quality parameters within the KB. Through this, effects of a particular water quality parameter can be propagated to related quality parameters (e.g. a reduction of parameter BOD will also cause a reduction of parameter COD).

5.2 Outlook and further research

The functionality of the software tool has been tested against a number of use cases. Some use cases have been derived based on characteristics of brewery wastewater taken from literature sources. A treatment configuration com-

monly applied for treatment of brewery wastewater as described in literature has been compared to results of the software tool.

From these tests the following three future task regarding the further development have been identified.

First, to prove the feasibility of the presented software tool its application within a real world scenario must be the next step. In this context it will be required to add new treatment units with additional technologies (e.g. chemical processes such as precipitation) to the KB. It is furthermore required to implement industry branch specific fractionation rules. Also cost information on treatment units must be adjusted to real world conditions.

Second, use case testing revealed limitations of the implemented search algorithm on initial flowsheets with more than three sources and sinks. To successfully address this problem it is essential to improve the MEA based search algorithm. This may be achieved by further implementation of backward search capabilities. Beyond this it may be useful to organize treatment technologies within the ontology according to their pre and post conditions in order to efficiently identify appropriate treatment options during the search process.

Third, the result of flowsheet generation being a list of ranked options, may not be useful for true practical applications. Instead it appears more appropriate to pre-process the results by a sensitivity analysis. Then feasible flowsheets can be evaluated in terms of robustness related to the variation of boundary conditions such as changing water quality parameters, energy cost etc.

Beyond the developed software tool this work shows promising possibilities and also limitations of an external logic based modeling framework codified in OWL(DL). From its original purpose, the semantic relation of web resources, OWL(DL) is designed for the representation of declarative knowledge (in contrast to procedural knowledge).

In that sense OWL(DL) can be regarded as a context free language which enables for intuitive representation of heterogeneous modeling knowledge within a single model framework, as long represented assertions are of pure declarative nature. In contrast the representation of complex modeling constructs, such as algebraic equations or logical conditions, is much more difficult since they need to be transferred into declarative assertions. The processing of these complex modeling constructs required furthermore a tailor made interpretation structure (e.g. equation parser).

It is the opinion of the author that the use of OWL(DL) for the purpose of model representation in engineering sciences will play an increasing role. This

is because the increasing complexity of modeling challenges across research disciplines requires a unified but expressive modeling language. OWL(DL) is a promising candidate for this purpose despite its limitations.

Glossary

adjacency list A data structure to describe a directed graph. Thereby a graph is described a list of its edges. Each edge is describe according to its state node and end node (Saake and Sattler, 2004).

asserted class taxonomy A taxonomy of a formal ontology as specified by the modeler before consistency checking (see inferred class taxonomy).

asserted state → state

axiom An assertion (including rules) in a logical form that together comprises the overall theory an ontology.

canonical modeling object Refers to a design pattern to represent single models in a standardized way to enable the forming of complexes models from single canonical modeling objects Bogusch *et al.* (2001, p. 971), Marquardt (1995b, p. 594).

class → concept

class hierarchy Organization of classes by sub- and super- class relations (syn.: taxonomy).

class restriction A restriction describes an anonymous class (syn.: unnamed class). The anonymous class contains all of the individuals that satisfy the restriction, i.e. all of the individuals that have the relationships required to be a member of the class (Horridge *et al.*, 2009).

class surrogate An individual that stands for a concepts to be able to link classes through properties (workaround for OWL(DL)).

closed world assumption (CWA) The set of objects and relations in a model includes everything necessary for the purpose of modeling (Kuipers 1994, p. 321, Russel and Norvig 2003, p. 439).

complete plan A plan which holds no open states or units. Can be represented by a graph where all nodes are linked by edges to other nodes.

concept Resembles abstract or real entities which have equal properties (syn.: class). In a broader sense a concept is a mental model of a abstract or real entity.

conceptual model related to declarative model

conceptual process design The first stage of within a design procedure. In this stage focus lies on the identification of process chains to achieve a given design objective.

conceptualization The process of identifying concepts and relations between them (capturing conceptual knowledge).

context property Describe properties of an overall wastewater state that is not based on a single wastewater constituent (e.g. temperature, pressure, pH value).

correct plan A plan where all preconditions of included units are fulfilled. Oppositely, an incorrect plan holds units whose preconditions are violated.

cyclic graph If there exists a path in a graph including a node more than once this graph is referred to as cyclic graph. (Luger, 2001).

declarative knowledge Declarative knowledge is defined as the factual information stored in memory and known to be static in nature. Other names, e.g. descriptive knowledge, propositional knowledge, etc. are also given. It is the part of knowledge which describes how things are. Things/events/processes, their attributes, and the relations between these things/events/processes and their attributes define the domain of declarative knowledge what (opposite: procedural knowledge).

design rationale An explicit documentation of the reasons behind decisions made when designing a system or artifact.

directed graph A graph wherein edges have a defined direction (defining successor and predecessor nodes) (Luger, 2001).

entity Something that exists by itself, although it need not be of material existence. In data modeling an entity is an unambiguous object that can be stored or processed. The object may be tangible or intangible, concrete or abstract.

epistemology The study of the origin, nature, and limits of human knowledge (Ceccaroni, 2001, p. 36).

expert system Expert systems are knowledge-based systems which emulate human reasoning using knowledge within a particular discipline (Heller *et al.*, 1998; Neumann, 2003).

extrinsic property Extrinsic properties (syn.: extensive property), such as volume and weight, are directly related to the amount of material being measured.

flow sheet A diagrammatic representation of the sequence of operations or equipment in an industrial process, computer program, etc.

fractionated state → state

goal state* Any plan that is correct and complete.

graph A set of nodes (N_i) and a set of edges (E_i) which relates pairs of nodes (Luger, 2001).

heuristic function, h(n) For a node n in search space, the heuristic function h(n) estimates the minimal distance between n and the final state.

hydraulic state Defines a state in terms of hydraulic parameter, see table 4.9.

individual Individuals are objects which cannot be divided without losing their structural and functional characteristics (Ceccaroni, 2001, p. 18) (syn.: instance, object, constant). In the context of OWL(DL) individuals are member of classes.

inference Engine → reasoner

inferred class taxonomy A taxonomy of a formal ontology resulting from consistency checking and reorganization of class hierarchy of defined classes by a reasoner (opposite: asserted class taxonomy).

initial plan A set of source and sink units (syn.: initial state*).

intrinsic property Intrinsic properties (syn.: intensive property) are those which are indepen-

dent of the quantity of matter present. For example, the density of gold is the same no matter how much gold you have to measure. Common intrinsic properties are density and specific gravity.

Issue-Based Information System (IBIS) A formal structure for the discussion and exploration of wicked problems (Bogusch *et al.*, 2001, p. 969).

JENA API A Java framework for building Semantic Web applications (see section 3.2).

material model Conceptual relationships between quality parameter for wastewater characterization through a formal representation.

Means-Ends Analysis (MEA) A strategy to control search in problem-solving. Given a current state and a goal state, an action is chosen which will reduce the difference between the two. The action is performed on the current state to produce a new state, and the process is recursively applied to this new state and the goal state.

model To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A (Minsky, 1965).

model environment A software program wherein a model is implemented and applied.

model framework Provides a modeling language and model concepts to transfer a mental or conceptual model into a model environment.

model unit Represents a device ore reactor. Consists of an logical and execution component. (see also process model)

Moving Bed Reactor (MBR) Biological wastewater treatment reactors where the biomass responsible for degradation is kept in suspensions either via an up-flow wastewater regime (in the case of anaerobic reactors for digestion) or by up-flow aeration system (in the case of aerobic oxidation).

ontology An ontology is a formal explicit specification of a shared conceptualization (Gruber, 1993b).

open state Any state within a plan which has at least one open successor unit.

open unit Any unit within a plan which has at least one open inlet port.

open world assumption (OWA) → Closed World Assumption.

parser In computing, a parser is one of the components in an interpreter or compiler that checks for correct syntax and builds a data structure (often some kind of parse tree, abstract syntax tree or other hierarchical structure) implicit in the input tokens.

path Within a graph the sequence of edges and nodes that connects two nodes is referred to as path. (Luger, 2001).

plan A set of units and wastewater states. Regarding state space search a plan is also referred to as state*.

port In the context of this work a generic model unit has one or two inlet ports as well as one or two outlet ports.

postcondition A qualitative estimation of how the incoming state will be changed by the model unit.

precondition A quantitative condition that must be fulfilled for that a model unit can be applied on a given state.

procedural knowledge Relates to knowledge of how to perform, or how to operate (syn.: know-how). (opposite: declarative knowledge)

process model An abstracted model of a real world phenomena for a distinct objective.

property A binary relation between two individuals (in OWL(DL)).

qualitative parameter Describe constituents contained in wastewater and are based on standardized measurement techniques (e.g. COD, TSS, N_{total}).

quantitative parameter Describe the hydraulic properties (flow) of a wastewater state.

reasoner A reasoner (or semantic reasoner) is a piece of software able to infer logical consequences from a set of asserted facts or axioms (syn.: inference engine).

reification The decision to represent a concept by class or an instance (Brachman and Levesque, 2004, p. 41). Reification make as-

sertions about assertions (Raffeiner, 2005, p. 53).

root graph A graph with a single node (N_0) which has no predecessor node (Luger, 2001).

search tree Is an abstract data structure wherein the set of elements which are to be search resemble to a tree like structure.

semantic As an element of language design semantic is devoted to the relation of signs to real world entities which they represent (Guizzardi, 2005) (i.e. meaning).

serialization format Serialization is about transferring data or knowledge into a format that can be stored (e.g. into a file). Serialization format is about the means for serialization. Types of serialization formats are for example XML, CSV.

settleable solids mass-concentration or volume part of undissolved particles, which settle in a given amount of time (DIN EN 1085) (Bischofsberger and Hegemann, 2000).

state A set of parameter value pairs that define the condition of a wastewater stream.

state space The set of all states* that can be reached from the initial state* through given transitions to possible goal states*.

state* In the context of state space search a state* refers to any possible condition that can be reached from the initial plan including correct/incorrect, complete/incomplete plans. (syn.: plan)

syntax As an element of language design syntax is devoted to the formal relation of signs to one another (Guizzardi, 2005) (i.e. grammar).

taxonomy → class hierarchy

tree A graph that has no cycles (Luger, 2001).

tuple An ordered list of elements.

W3C The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the Web.

Nomenclature

A area (m^2)

C_{invest} investment cost

$C_{running}$ running or variable cost

C concentration $(g/m^3$ or $mg/l)$

d diameter

g gravity

h(n) heuristic function of node n.

hs hydraulic state

i individual/inert

K half saturation constant

L load

$\hat{\mu}$ growth rate

Q_h flow rate (m^3/h)

Q_r recycle flow rate (m^3/h)

Q_w flow rate of excess sludge (m^3/h)

ρ_f fluid density

ρ_p particle density

ρ reaction kinetics

R recycle ratio

s_i* initial state*

$s*$ a state* in state space (syn.. a plan)

S soluble

s state of a waste stream

V volume (m^3)

v_s settling velocity (m/s)

X particulate

Bibliography

AbwV (2004). Verordnung über Anforderungen an das Einleiten von Abwasser in Gewässer (AbwV), Anhang 1, Häusliches und kommunales Abwasser. Bundesgesetzblatt Nr. 49 vom 28.9.2001.

L. Akça, C. Kinaci, and M. Karpuzcu (1993). A Model for optimum design of activated sludge plants. *Water*, **27**(9):1461–1468.

S. Ali (1999). *Synthesis of Batch Processing Schemes for the Production of Pharmaceuticals and Specialty Chemicals*. Ph.D. thesis, Department of Chemical Engineering, Massachusetts Institute of Technology.

A. Alva-Argáez, A. Kokossis, and R. Smith (1998). Wastewater minimization of industrial systems using an integrated approach. *Computers & Chemical Engineering*, **22**:S741–S744.

C. Angeli (2007). Theoretische Grundlagen des Semantic Web. Technical report, Universität Augsburg, Institut für Informatik, Programmierung verteilter Systeme.

G. Antoniou and F. van Harmelen (2004). *A Semantic Web Primer*. MIT Press.

J. Arizmendi-Sánchez and P. Sharratt (2005). Multilevel phenomenological modelling approach to support the evaluation and generation of intensified processes. *Computer Aided Chemical Engineering*, **20**:901–906.

J. Arizmendi-Sanchez and P. Sharratt (2008). Phenomena-based modularisation of chemical process models to approach intensive options. *Chemical Engineering Journal*, **135**:83–94.

V. Ashley and P. Linke (2004). A Novel Approach for Reactor Network Synthesis using knowledge discovery and optimization techniques. *Chemical Engineering Research and Design*, **82(A8)**:952–960.

D. Augustin, J. Strauss, B. Fineberg, M. Johnson, R. Linebarger, and F. Sansom (1967). The SCi Continuous System Simulation Language (CSSL).

Simulation, **9**:(6).

F. Baader (1996). Logic-based Knowledge Representation. *KI*, **3:96**:8–16.

F. Baader, I. Horrocks, and U. Sattler (2003). *Lecture Notes in Computer Science*, chapter Description Logics as ontology languages for the semantic web. Springer-Verlag.

M. Bagajewicz (2000). A review of recent design procedures for water networks in refineries and process plants. *Computers & Chemical Engineering*, **24**:2093–2113.

M. Baumeister (2000). *Ein Objektmodell zur Repräsentation und Wiederverwendung verfahrenstechnischer Prozessmodelle*. Ph.D. thesis, Fakultät Maschinenwesen, RWTH Aachen.

B. Bayer and W. Marquardt (2003). A Comparision of Data Models in Chemical Engineering. *Concurrent Engineering: Research and Applications*, **11**:129.

B. Bayer, R. Schneider, and W. Marquardt (2000). Integration of data models for process design - first steps and experience. *Computers & Chemical Engineering*, **24**:599–605.

J. Bieszczad (2000). *A Framework for the Language and Logic of Computer-Aided Phenomena-Based Process Modeling*. Ph.D. thesis, Department of Chemical Engineering - Massachusetts Institute of Technology.

W. Bischofsberger, N. Dichtl, K. Rosenwinkel, C. Seyfried, and B. Böhnke (2005). *Anaerobtechnik*. Springer-Verlag Berlin Heidelberg.

W. Bischofsberger and W. Hegemann (2000). *Lexikon der Abwassertechnik*. Vulkan-Verlag Essen.

E. Blomqvist (2009). *Semi-automitic Ontology Construction based on Patterns*. Ph.D. thesis, Department of Computer and Information Science Linköping Universitet, Linköping, Sweden.

R. Bogusch, B. Lohmann, and W. Marquardt (2001). Computer-aided process modeling with MODKIT. *Computers & Chemical Engineering*, **25**:963–995.

R. Bogusch and W. Marquardt (1997). A formal representation of process model equations. *Computers & Chemical Engineering*, **21**:1105–1115.

C. Boyle and B. Baetz (1998). A prototype knowledge-based decision support system for industrial waste management: part I. The decision support sys-

tem. *Waste Management*, **18**:87–97.

R. Brachman and H. Levesque (2004). *Knowledge Representation and Reasoning*. Elsevier.

J. Brase (2005). *Usage of metadata*. Ph.D. thesis, Fakultät für Elektrotechnik und Informatik der Universität Hannover.

B. Braunschweig and R. Gani (2002). *Software Architectures and Tools for Comuter Aided Process Engineering*, chapter Introduction, pages 3–16. Elsevier Science B.V.

R. Butner (1999). A heuristic design advisor for incorporating pollution prevention concepts in chemical process design. *Clean Products and Processes*, **1**:164–169.

L. Ceccaroni (2000). Integration of a rule-based expert system, a case-based reasoner and an ontological knowledge -base in the wastewater domain. In *Proceedings of 2nd ECAI Workshop on Binding Environmental Sciences and AI (BESAI2000)*.

L. Ceccaroni (2001). *ONTOWEDSS - An ontology-based environmental decision-support system for the management of wastewater treatment plants*. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain.

A. Chakraborty and A. Linninger (2002). Plant-Wide Waste Management. 1. Synthesis and Multiobjective Design. *Industrial & Engineering Chemistry Research*, **41**:4591–4604.

R. Chakraborty (2010). Problem Solving, Search and Control Strategies. Technical report, Dept. of Computer Science & Engineering, Jaypee University of Engineering and Technology (JUET), Guna.

D. Chapman (1987). Planning for Conjunctive Goals. *Artifical Intelligence*, **32**:333–377.

C. Crooks and S. Macchietto (1992). A combined MILP and logic-based approach to the synthesis of operating procedures for batch plants. *Chemical Engineering Communications*, **114**:117–144.

L. d'Anterroches (2005). *Process Flowsheet Generation & Design through a Group Contribution Approach*. Ph.D. thesis, Department of Chemical Engineering Technical University of Denmark.

R. Darton, R. Prince, and D. Wood, editors (2003). *Chemical Engineering: Vision of the World*. Elsevier Science B.V.

R. Davis, H. Schrobe, and P. Szolovitz (1993). What is a knowledge representation? *AI Magazin*, **14:1**:17–33.

J. de Bruijn (2003). Using Ontologies - Enabling Knowledge Sharing and Reuse on the Semantic Web. Technical report, DERI - Digital Enterprise Research Intstitute - DERI Technical Report DERI-2003-10-29.

E. Delekgurgen, S. Doğruel, O. Karahan, and D. Orhon (2006). Size distribution of wastewater COD fractions as an index for biodegradability. *Water Research*, **40**:273–282.

R. Dick and K. Young (1972). Analysis of thickening performance of final settling tanks. In *27th Ind. Waste Conf., Purdue Univ., Lafayette, Ind.(May 1972)*.

D. Dochain and P. Vanrolleghem (2001). *Dynamical modelling and estimation in wastewater treatment processes*. IWA Publishing.

J. Douglas (1985). A hierarchical decision procedure for process synthesis. *AICHE Journal*, **31**:353.

J. Douglas (1988). *Conceptual Design of Chemical Processes*. McGraw-Hill chemical engineering series.

J. Douglas (1995). Synthesis of Separation System Flowsheets. *AIChE Journal*, **41**(12):2522–2536.

T. Drengstig, S. Wasbø, and B. Foss (1997). A formal graphical based process modeling methodology. *Computers & Chemical Engineering*, **21**:835–840.

DWA (2009). *Industriewasserbehandlung*. Verlag der Bauhaus Universität Weimar.

M. Eggersmann, J. Hackenberg, W. Marquardt, and I. Cameron (2002). *Software Architectures and Tools for Computer Aided Process Engineering*, chapter Applications of Modelling - A Case Study from Process Design, pages 335–372. Elsevier Sciences B.V. Braunschweig, B. AND Gani, R.

H. Elmqvist, D. Brück, and M. Otter (1996). Dymola - User's Manual. Technical report, Dynasim AB, Lund, Sweden.

E. Evenson and B. Baetz (1994). Selection and Sequencing of hazardous waste treatment processes: A knowledge based systems approach. *Waste Manage-*

ment, **14**:161–165.

M. Fernándes-López and Gómez-Pérez (2000). OntoWeb; Deliverable 1.4: A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies. Technical report, Vrije Universiteit Amsterdam, the Netherlands.

X. Flores Alsina (2008). *Conceptual Design of Wastewater Treatment Plants using multiple objectives*. Ph.D. thesis, Universitat de Girona, Spain.

I. Freitas, A. Carlos, A. Costa, and R. Boaventura (2000). Conceptual design of industrial wastewater treatment processes: primary treatment. *Computers & Chemical Engineering*, **24**:1725–1730.

R. Gani and I. Papaeconomou (2006). *Batch Processes*, chapter Conceptual Design and Synthesis of Batch Processes, pages 43–82. CRC Press. Korovessi, E. AND Linninger, A.A.

W. Geiger and I. Nafo (1999). Grundlagen der Wasserversorgung und der Abwassertechnik. Technical report, Universität Essen Siedlungswasserwirtschaft.

D. Giokas, Y. Kim, P. Paraskevas, E. Paleologos, and T. Lekkas (2002). A simple empirical model for activated sludge thickening in secondary clarifiers. *Water Research*, **36**(13):3245–3252.

B. Gleich (2008). Ontologie-Methodologien und Ontologie Design. Technical report, Universität Augsburg.

V. Gold, K. Loening, A. McNaught, and P. Sehmi (1987). *Compendium of Chemical Terminology*. Blackwell, Oxford.

K. Grijspeerdt, P. Vanrolleghem, and W. Verstraete (1995). Selection of one-dimensional sedimentation: models for on-line use. *Water Science & Technology*, **31**(2):193–204.

T. Gruber (1993a). *Formal Ontology in Conceptual Analysis and Knowledge Representation*, chapter Towards Principles for the Design of Ontologies Used for Knowledge Sharing. Kluwer Academic Publishers. Guarino, N. AND Poli, R.

T. Gruber (1993b). A translation approach to portable ontology specifications. *Knowledge Acquisition*, **5**:199–220.

T. Gruber *et al.* (1995). Toward principles for the design of ontologies used for knowledge sharing. *International journal of human computer studies*, **43**(5):907–928.

M. Gruninger, M. Fox *et al.* (1995). Methodology for the Design and Evaluation of Ontologies. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI*, volume 95.

N. Guarino (1998). Formal Ontology and Information Systems. In *Formal Ontology in Informations Systems, Trento, Italy, 6-8 June 1998.*.

N. Guarino and C. Welty (2002). Evaluating ontological decisions with OntoClean. *Communications of the ACM*, **45**(2):61–65.

N. Guarino and C. Welty (2009). An overview of OntoClean. *Handbook on ontologies*, pages 201–220.

G. Guizzardi (2005). *Ontological Foundations for Structural conceptual Models*. Ph.D. thesis, Centre for Telemeatics and Information Technology, Enschede, The Netherlands.

W. Gujer (2008). *Systems Analysis for Water Technology*. Springer Verlag.

M. A. Hamouda, W. B. Anderson, and P. M. Huck (2009). Decision support systems in water and wastewater treatment process selection and design: a review. *Water Science & Technology*, **60**(7):1757–1770.

K. Hangos and I. Cameron (2001). *Process Modelling and Model Analysis*. Academic Press.

S. Hansen, C. Crosby, and K. Dostal (1988). Epa treatability database. Technical report, Risk Reduction Engineering Laboratory, USEPA, Cincinnati (OH).

G. Harmsen (2004). Industrial best practices of conceptual process design. *Chemical Engineering and Processing*, **43**:677–681.

M. Heller, S. Garlapati, and K. Aithala (1998). Expert membrane system design and selection for metal finishing waste water treatment. *Expert Systems with Applications*, **14**(3):341 – 353.

M. Henze, C. Grady, W. Gujer, G. Marais, and T. Matsuo (1987). A general model for single -sludge wastewater treatment systems. *Water Research*, **21(5)**:505–517.

M. Henze, W. Gujer, T. Mino, and M. van Loosdrecht (2000). Activated Sludge Models ASM1, ASM2, ASM2d and ASM3. Technical report, IWA Publishing.

M. Henze, van Loosdrecht M.C.M., G. Ekama, and D. Brdjonovic, editors (2008). *Biological Wastewater Treatment*. IWA Publishing.

G. Henüen (2004). *Kostenoptimale Gestaltung von Stoffaustauschernetzwerken mit Hilfe der erweiterten Wasser-Pinch-Methode*. Ph.D. thesis, Fakultät für Maschinenwesen, RWTH-Aachen.

M. Herb (2006). Ontology Engineering mit OntoClean. *IPD University Karlsruhe*, **10**:2011.

D. Hocking, J. Nougués, J. Rodríguez, and S. Sama (2002). *Software Architectures and Tools for Comuter Aided Process Engineering*, chapter Simulation, Design & Analysis, pages 165–191. Elsevier Science B.V. Braunschweig, B. AND Gani, R.

R. Hoekstra (2009). *Ontology Representation - Design Patterns and Ontologies that make sense*. Ph.D. thesis, Dutch Research School for Information and Knowledge Systems.

M. Hofmeister (1998). BatchKit - A knowledge integration environment for process engineering. *Computers & Chemical Engineering*, **22**:109–123.

M. Horridge, N. Drummond, J. Goodwin, A. Rector, R. Stevens, and H. Wang (2006). The manchester owl syntax. *OWL: Experiences and Directions*, pages 10–11.

M. Horridge, S. Jupp, G. Moulton, A. Rector, R. Stevens, and C. Wroe (2009). A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition1. 2. *The University of Manchester*.

M. Hostrup, P. M. Harper, and R. Gani (1999). Design of environmentally benign processes: integration of solvent design and separation process synthesis. *Computers & Chemical Engineering*, **23**:1395–1414.

M. Hunze (2005). *Simulation in der kommunalen Abwasserreinigung*. Oldenbourg Industrieverlag.

S. Hölldobler (2003). *Logik und Logikprogrammierung*. SYNCHRON Heidelberg.

K. Imhoff and K. Imhoff (1993). *Taschenbuch der Stadtentwässerung*. Oldenbourg Verlag München Wien.

C. A. Jaksland, R. Gani, and K. M. Lien (1995). Separation process design and synthesis based on thermodynamic insights. *Chemical Engineering Science*, **50**(3):511 – 530. ISSN 0009-2509.

G. Klir (2001). *Facets of systems sciences*. Kluwer Academic/Plenum Publisher, New York.

G. Klir and D. Elias (2003). *Systems Problem Solving*. Kluwer Academic Publishers.

P. Koppe and A. Stozek (1993). *Kommunales Abwasser - Seine Inhaltsstoffe nach Herkunft, Zusammensetzung und Reaktion im Reinigungsprozess einschließlich Klärschlamm*. Vulkan Verlag.

B. Kuipers (1994). *Qualitative Reasoning - Modeling and Simulation with Incomplete Knowledge*. The MIT Press.

U. Leinweber (2002). *Anforderungen an die integrierte Modellierung von Entwässerungssystem und Kläranlage*. Ph.D. thesis, Universität Kaiserslautern.

P. Lessard and M. Beck (1993). Dynamic modelling of the activated sludge process: a case study. *Water Research*, **27**(6):963 – 978. ISSN 0043-1354.

A. D. Levine, G. Tchobanoglous, and T. Asano (1991). Size distributions of particulate contaminants in wastewater and their impact on treatability. *Water Research*, **25(8)**:911–922.

D. Lewis (1986). *On the Plurality of Worlds*. Blackwell Publishing.

X. Li (2004). *Conflict-based Method for Conceptual Process Synthesis*. Ph.D. thesis, University of Technology Lappeeranta, Finnland.

X. Li and A. Kraslawski (2004). Conceptual process synthtesis: past and current trends. *Chemical Engineering and Processing*, **43**:589–600.

P. Linke and A. Kokossis (2003). Advanced process systems design technology for pollution prevention and waste treatment. *Advances in Environmental Research*, **8**:229–245.

A. Linninger (2001). Recent Advances in Process Systems Engineering. In *IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary, May 21-23*.

A. Linninger, S. Ali, and G. Stephanopoulos (1996a). Knowledge-based validation and waste management of batch pharmaceutical process designs. *Computers & Chemical Engineering*, **20**:1431–1436.

A. Linninger and A. Chakraborty (1999). Synthesis and optimization of waste treatment flowsheets. *Computers & Chemical Engineering*, **23**:1415 – 1425.

A. Linninger, A. Chakraborty, and R. Colberg (2000a). Planning of waste reduction strategies under uncertainty. *Computers & Chemical Engineering*, **24**:1043–1048.

A. Linninger, S. Chowdhry, V. Bahl, H. Krendl, and H. Pinger (2000b). A systems approach to mathematical modeling of industrial processes. *Computers & Chemical Engineering*, **24**:591–598.

A. Linninger, M. Hofer, H. Krendl, H. Druckenthaner, and H. Jörgl (1996b). M-Projekt - Organizing Problem Representation and Modeling of steady state and dynamic processes. *Computers & Chemical Engineering*, **20**:425–430.

A. Linninger, E. Stephanopoulos, S. Ali, C. Han, and G. Stephanopoulos (1995). Generation and Assessment of Batch Processes with Ecological Considerations. *Computers & Chemical Engineering*, **19**:7–13.

A. Linninger and G. Stephanopoulos (1998). A Natural Language Approach for the design of Batch Operating Procedures. *Informatica*, **22**:423–434.

B. Lohmann and W. Marquardt (1996). On the Systemization of the Process of Model Development. *Computers & Chemical Engineering*, **20**:213–218.

G. Luger (2001). *Künstliche Intelligenz*. Pearson Studium.

M. Mangold, S. Motz, and E. D. Gilles (2002). A network theory for the structured modelling of chemical processes. *Chemical Engineering Science*, **57**(19):4099 – 4116. ISSN 0009-2509.

W. Marquardt (1992). Rechnergestützte Erstellung verfahrenstechnischer Prozeßmodelle. *Chemie Ingenieur Technik*, **64**:(1) 25–40.

W. Marquardt (1995a). *Methods of Model Based process Control*, chapter Towards a process modelling methodology, pages 3–40. Kluwer Academic Publishers. Berber, R.

W. Marquardt (1995b). Trends in Computer-Aided Process Modelling. *Computers & Chemical Engineering*, **20**:591–609.

W. Marquardt (2005). Modellbildung und Analyse verfahrenstechnischer Prozesse. Technical report, Lehrstuhl für Prozesstechnik, RWTH-Aachen.

W. Marquardt, J. Morbach, A. Wiesner, and A. Yang (2010). *OntoCAPE - A Re-Usable Ontology for Chemical Process Engineering*. Springer Verlag.

D. MCDermott and J. Hendler (1995). Planning: What it is, What it could be, And introduction to the Special Issue on Planning and Scheduling. *Artificial Intelligence*, **76**:1–16.

D. McGuiness (2003). *Ontologies Come of Age*. MIT Press.

M. Minsky (1965). Models, Minds, Machines. In *Proceedings of the IFIP Congress, pages 45-49*.

Mitchell and Gouthier Ass. (1992). Advanced Continuous Simulation Language (ACSL). Beginners Guide. Technical report, Mitchell and Gouthier Ass.

R. Mizoguchi (2004a). Part 3: Advanced course of ontological engineering. *New Generation Computing*, **22**(2):193–220.

R. Mizoguchi (2004b). Tutorial on ontological engineering part 2: Ontology development, tools and languages. *New Generation Computing*, **22**(1):61–96.

Modelica Association (2000). Modelica-A Unified Object-Oriented Language for Physical Systems models. Language Specification. Technical report, Modelica Association.

R. Molitor (2000). *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken*. Ph.D. thesis, Fakulät für Mathematik, Informatik und Naturwissenschaften; RWTH Aachen.

J. Morbach (2009). *A Reusable Ontology for Computer-Aided Process Engineering*. Ph.D. thesis, RWTH Aachen University, Fakultät Maschinenwesen, Lehrstuhl Prozesstechnik.

J. Morbach, B. Bayer, A. Yand, and W. Marquardt (2008a). *Collaborative and Distributed Chemical Engineering, LNCS 4970*, chapter Product Data Models, pages 93–110. Springer Verlag. Nagl, M. AND Marquardt, W.

J. Morbach, M. Theißen, and W. Marquardt (2008b). *Collaborative and Distributed Chemical Engineering, LNCS 4970*, chapter An Introduction to Ap-

plication Domain Modelling, pages 83–92. Springer Verlag. Nagl, M. AND Marquardt, W.

J. Morbach, A. Wiesner, and W. Marquardt (2007a). A Meta Model for the Design of Domain Ontologies. Technical report, RWTH Aachen University, Aachener Verfahrenstechnik, Process Systems Engineering.

J. Morbach, A. Yang, and W. Marquardt (2007b). OntoCAPE - A large-scale ontology for chemical process engineering. *Engineering Applications of Artifical Intelligence*, **20**:147–167.

J. Morbach, A. Yang, A. Wiesner, and W. Marquardt (2008c). Supporting Concepts (LPT-2008-26). Technical report, RWTH Aachen University Aachener Verfahrenstechnik Process Systems Engineering.

J. Morbach, Y. Yang, and W. Marquardt (2008d). Material (Technical Report LPT-2008-27). Technical report, RWTH Aachen University Aachener Verfahrenstechnik Process Systems Engineering.

M. Neumann (2003). Ökologische Informatik - Vielältige Forschung mit Bezug zur Geoökologie. *Forum Geoökologie*, **14(1)**:4–6.

B. Nilsson (1993). *Object-Oriented Modeling of Chemical Processes*. Ph.D. thesis, Lund Institute of Technology, Sweden.

N. Noy and D. McGuinness (2005). Ontology Development 101: a Guide to Creating your first Ontology. Technical report, Stanford University.

G. Olsson and B. Newell (1999). *Wastewater Treatment Systems - Modelling, Diagnosis and Control*. IWA Publishing.

P. Patel-Schneider (2004). What is OWL (and why should I care)? In *Ninth International Conference on the Principles of Knowledge Representation and Reasoning, Whistler, Canada, June 2004*.

E. E. Petersen (1965). *Chemical Reaction Analysis*. Prentice Hall.

M. Poch, J. Comas, R.-R. I., M. Sànchez-Marré, and C. U. (2004). Designing and building real environmental decision support systems. *Environmental Modelling & Software*, **19**(9):857 – 873. Environmental Sciences and Artificial Intelligence.

Process Systems Enterprise (1997). gPROMS Introductory User Guide. Technical report, Process Systems Enterprise, London, UK.

L. Puigjaner, M. Graells, and G. Reklaitis (2002). *Software Architectures and Tools for Computer Aided Process Engineering*, chapter Computer Tools for Discrete/Hybrid Production Systems, pages 213–228. Elsevier Science B.V.

S. Raffeiner (2005). *Modelling Ontolgies with Topic Maps and OWL: Implentation Challenges and Conceptual Issues*. Master's thesis, Institut für Softwaretechnik und interaktive Systeme der Technischen Universität Wien.

R. Rautenbach (1997). *Membranverfahren, Grundlagen der Modul- und Anlagenauslegung*. Springer Verlag.

I. Reimann, A. Klatt, and H. Märkl (2002). Behandlung fetthaltiger Abwässer der Lebensmittelindustrie mit einem thermophilen Mikororganismus. *Chemie Ingenieur Technik*, **74**:508 – 512.

U. Reinhardt (1995). *Erweiterung des objektorientierten verfahrenstechnischen Datenmodells VeDa um Repräsentationskonzepte für Perspektiven, Versionen und Hierarchien*. Master's thesis, Lehrstuhl für Prozeßtechnik, RWTH Aachen.

D. Rickert and J. Hunter (1971). General Nature of Soluble and Pariclulate Organics in Sewage and Secondary Effluent. *Water Research*, **5**:421–436.

S. Rigopoulos and P. Linke (2002). Systematic development of optimal activated sludge process design. *Computers & Chemical Engineering*, **26**:585–597.

I. Roda, M. Poch, and R. Bãnares Alcántara (2000a). Application of a support system to the design of wastewater treatment plants. *Artifical Intelligence in Engineering*, **14**:45–61.

I. Roda, M. Poch, and R. Bãnares Alcántara (2000b). Conceptual design of wastewater treatment plants using a design support system. *Journal of Chemical Technology and Biotechnology*, **75**:73–81.

I. Roda, M. Sánchesz-Marré, J. Comas, J. Baeza, J. Colprim, J. Lafuente, U. Cortés, and P. M. (2002). A hybrid supervisory system to support WWTP operation: implementation and validation. *Water Science & Technology*, **45**:(4–5) 289–297.

M. Rodriguez (2005). A new algorithm to develop hybrid systems. *Computers & Chemical Engineering*, **30**:260–267.

G. Rotstein, R. Lavie, and D. Lewin (1994). Automatic Synthesis of Batch Plant Procedures: Process-Oriented Approach. *AIChE Journal*, **40**(10):1650–1664.

S. Russel and P. Norvig (2003). *Artificial Intelligence a Modern Approach*. Prentice Hall.

H. Rüffer and K. Rosenwinkel, editors (1985). *Lehr-und Handbuch der Abwasser Technik: Band V - Organisch verschmutzte Abwässer der Lebensmittelindustrie*. Ernst & Sohn Verlag für Architektur und technische Wissenschaften, Berlin.

G. Saake and K.-U. Sattler (2004). *Algorithmen und Datenstrukturen*. dpunkt-Verlag.

A. Samantray and B. Bouamama (2008). *Model-based Process Supervision*. Springer Verlag.

M. Schmidt-Schauss and G. Smolka (1991). Attributive concept description with complements. *Artifical Intelligence*, **48**:1–26.

Schwister, K. ed. (2003). *Taschenbuch der Umwelttechnik*. Fachbuchverlag Leipzig im Carl Hanser Verlag.

T. Seuranen (2006). *Studies on Computer-aided conceptual process design*. Ph.D. thesis, Helsinki Universtiy of Technology.

J. Siirola and D. Rudd (1971). Computer-Aided Synthesis of Chemical Process Designs - From Reaction Path Data to the Process Task Network. *Ind. Eng. Chem. Fundam.*, **10**:353–363.

M. Sipser (1998). *Introduction to the Theory of Computation*. PWS Publishing, Boston.

R. Smith (1969). Preliminary design of wastewater treatment systems. *J Sanit Engng Div Am Soc Civ Engrs*, **95**:117–145.

C. Sophonsiri and E. Morgenroth (2004). Chemical composition associated with different particle size fractions in municipal, industrial, and agricultural wastewaters. *Chemosphere*, **55**:691–703.

F. Sowa, J (2000). *Knowledge representation: logical, philosophical and computational foundations*. Pacific Grove, Brooks/Cole, CA.

W. Stahl (2004). *Industrie Zentrifugen - Band II - Maschinen und Verfahrenstechnik,*. DrM Press.

A. Stankiewicz (2002). Process intensification. *Ind. Eng. Chem. Res.*, **41**:1920–1924.

G. Statyukha, O. Kvitka, I. Dzhygyrey, and J. Jezowski (2008). A simle sequential approach for designing industrial wastewater treatment networks. *Journal of Cleaner Production*, **16**:215–224.

P. Steele (1999). Modelling Frameworks: The Essential Link Between Models and Methodologies. In *Proceedings 10th Australasian Conference on Information Systems*.

B. Stein (2008). Model Construction for Knowledge-Intensive Engineering Tasks. In Y. Liu, A. Sun, H. T. Loh, W. F. Lu, and E.-P. Lim, editors, *Advances of Computational Intelligence in Industrial Systems*, volume 116, chapter 7, pages 139–167. Springer Verlag, Berlin, Heidelberg.

G. Stepanopoulos, G. Henning, and H. Leone (1990a). MODEL.LA A modeling language for process engineering - I: The formal framework. *Computers & Chemical Engineering*, **14**:(8) 813–846.

G. Stepanopoulos, G. Henning, and H. Leone (1990b). MODEL.LA A modeling language for process engineering - II: Multifaceted modeling of processing systems. *Computers & Chemical Engineering*, **14**:(8) 847–869.

R. Sykes (2003). *The Civil Engineering Handbook, second edition*, chapter Biological Wastewater Treatment. CRC Press LLC.

J. Tanskanen, V. Pohjola, and K. Lien (1995). Phenomenon driven process design methodology: Focus on reactive distillation. *Computers & Chemical Engineering*, **19**(Supplement 1):77–82. ISSN 0098-1354.

T. Tebbutt and D. Christoulas (1975). Performance relationships for primary sedimentation. *Water Research*, **9**(4):347 – 356. ISSN 0043-1354.

S. Tessaris (2001). *Questions and Answers: Reasoning and Querying in Description Logic*. Ph.D. thesis, University of Manchaster, Faculty of Science and Engineering.

M. Uschold and M. King (1995). Towards a methodology for building ontologies. In *Workshop on basic ontological issues in knowledge sharing*, volume 74. Montreal, Canada.

S. Vaillant, M.-F. Pouet, and O. Thomas (1999). Methodology for the characterisation of heterogeneous fractions in wastewater. *Talanta*, **50**:729–736.

W. Verstraete and S. Vlaeminck (2011). ZeroWasteWater: short-cycling of wastewater resources for sustainable cities of the future. *International Journal of Sustainable Development & World Ecology*, **18**(3):253–264.

P. Vesilind (1968). *The influence of stirring in the thickening of biological sludge*. Ph.D. thesis, University of North Carolina, Chapel Hill.

S. Walter (2005). *Untersuchung verfahrenstechnischer Möglichkeiten zur Brauchwasserkreislaufführung in der Brauerei*. Ph.D. thesis, Fakultät Wissenschaftszentrum Weihenstephan für Ernährung, Landnutzung und Umwelt der Technische Universität München.

S. Wasbøand B. Foss (1996). Modelling unit processes using formal language description and object orientation. *Mathematical Modelling of Systems*, **1**:1–30.

L. v. Wedel, W. Marquardt, and R. Gani (2002). *Software Architectures And Tools For Comuter Aided Process Engineering*, chapter Modelling Frameworks, pages 89–125. Elsevier Sciences B.V. Braunschweig, B. AND Gani, R.

D. Weld (1999). Recent advances in AI planning. *AI magazine*, **20**(2):93.

A. Westerber (1989). Synthesis in engineering design. *Computers & Chemical Engineering*, **13 (4-5)**:365–376.

W. Wukovits, M. Harasek, and A. Friedl (2003). A knowledge based system to support the process selection during wastewater treatment. *Resources, Conservation and Recycling*, **37**:205–215.

A. Yang, B. Schlüter, B. Bayer, J. Krüger, E. Haberstroh, and W. Marquardt (2003). A concise conceptual model for material data and its applications in process engineering. *Computers & Chemical Engineering*, **27**:595 – 609.

A
On organic carbon and nutrient removal

A.1 Dimensioning of aeration tanks according to Henze *et al.* (2008)

This section describes the dimensioning of aeration tanks for organic carbon removal according to Henze *et al.* (2008).
For the application within the sequential planning algorithm the model unit must provide:

1. a description for the state transition of the incoming state (inflow), i.e. processes on carbon removal (open for extension for nitrogen removal)
2. the recycle flow Q_R (for the succeeding separation and split unit)
3. the flow on excess sludge Q_W (for the succeeding separation and split unit)

Required design parameters may be calculated depending on the inflow or preliminary defined (e.g. recycle ratio R, sludge retention time SRT, reactor volume V).
The following presumption is thereby accepted slowly biodegradable organics are completely utilized (sludge age more than 3 days). Hence no distinction between slowly and readily biodegradable (solid, suspended or particulate) organics are made Henze *et al.* (2008, p. 57 ff.).
The inflow concentration on total organic is represented by C^{in}_{total} in $gCOD/m^3$ measured by COD_{total}. The following fractions are derived therefrom:

- biodegradable organics load L^{in}_B n $gCOD/m^3$ (eq. A.2)
- un-biodegradable particulate organic load $L^{in}_{UB,X}$ n $gVSS/m^3$

- inorganic suspended solids (ISS) load $L^{in}_{IO,X}$ n $gISS/m^3$

$$
\begin{aligned}
L^{in}_{total} &= Q^{in} C^{in}_{total} && \text{(A.1)}\\
L^{in}_{B} &= Q^{in} C^{in}_{B} && \text{(A.2)}\\
&= Q^{in}(C^{in}_{S,B} + C^{in}_{X,B})\\
&= L^{in}_{total}(1 - f_{us} - f_{up})\\
L^{in}_{UB,X} &= Q^{in} S^{in}_{total} \cdot \frac{f_{up}}{f_{cv}} && \text{(A.3)}\\
&= L^{in}_{total} \cdot \frac{f_{up}}{f_{cv}}\\
L^{in}_{IO} &= Q^{in} C^{in}_{IO} && \text{(A.4)}
\end{aligned}
$$

The objective is to determine the required reactor volume, V for the activated sludge process. For the determination of the reactor volume the following assumptions have been defined:

- settling velocity index, SVI=150 l/kg
- specific sludge production, s = 1 kgTS/kgBOD
- thickening time, t_e=1 h
- recycle ratio, R=1

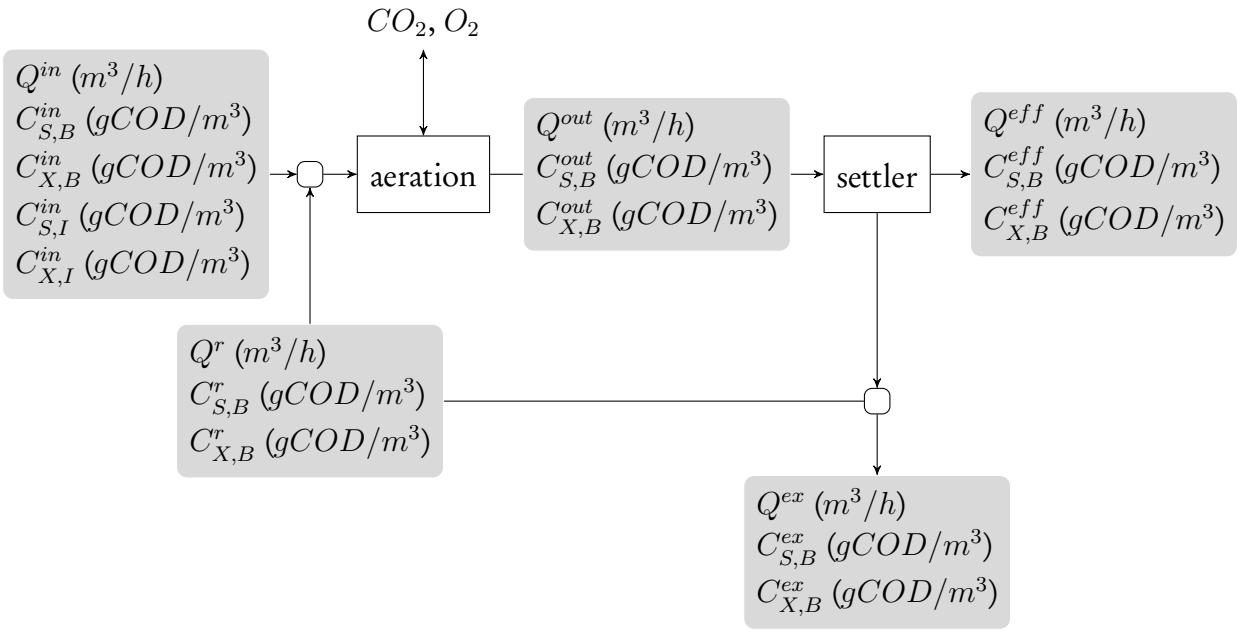


Figure A.1: Notation of fractions in the MBR cycle according to Henze *et al.* (2008)

- sludge age, SRT=4 d

Given the inflow concentration of biodegradable substrate C^{in}_{BOD} in g/m^3 the reactor volume in m^3 can be calculated as follows:

$$C_{TSS,sludge} = \frac{1000}{SVI} \cdot t_e^{1/3} \quad \text{(A.5)}$$

$$C_{TSS,recycle} = 0.7 \cdot C_{TSS,sludge} \quad \text{(A.6)}$$

$$C_{TSS,reactor} = \frac{R \cdot C_{TSS,recycle}}{(1+R)} \quad \text{(A.7)}$$

$$L_{ex,24} = s \cdot Q_{24} \cdot C^{in}_{BOD} \quad \text{(A.8)}$$

$$V = \frac{SRT \cdot L_{ex,24}}{C_{TSS,reactor}} \quad \text{(A.9)}$$

In the above equations $L_{ex,24}$ stands for the excess sludge load per day ($kgTS/d$). Following the assumptions $TS_{reactor}$ relates to 2.3 $kgTS/m^3$.

Alternative may the daily sludge production s in $gTSS/gCOD$ be calculated as follows (Henze *et al.*, 2008, p. 72):

$$\frac{L_{TSS}}{L^{in}_{total}} = \frac{1}{f_i}\left(\frac{(1-f_{us}-f_{up})Y_H}{(1+b_H SRT)} \cdot (1+f_H b_H SRT) + \frac{f_{up}}{f_{cv}}\right) \quad \text{(A.10)}$$

Table A.1: Modeling parameters Henze *et al.* (2008, p. 35)

Constant	**Symbol**	**Standard value at 20 °C**
un-biodegradable soluble COD fraction (-)	f_{us}	-
un-biodegradable particulate COD fraction (-)	f_{ux}	-
yield coefficient (gCOD/gCOD)	Y_H	0.61
yield coefficient (gVSS/gCOD)	Y_{Hv}	0.45
endogenous respiration rate* (1/d)	b_H	0.24
endogenous residue fraction (-)	f_H	0.2
ISS content of OHOs (-)	f_{iOHO}	0.15
COD to VSS ratio of sludge (gCOD/gVSS)	f_{cv}	1.48

Table A.2: Exemplary wastewater concentrations (in mg/l), Henze *et al.* (2008, p. 35)

Parameter	Amount
COD total	1200
COD soluble	480
COD suspended	720
BOD	560
N total	100
Ammonia-N	75
P total	25
Ortho-P	15
TSS	600
VSS	480

Example
Given an hourly inflow of Q^{in} of 10 m^3/h and the wastewater characteristic denoted in table A.2 the volume is calculated as follows (equation A.9):

$$\begin{aligned} L_{ex,24} &= 1kgTS/kgBOD \cdot 2400m^3/d \cdot 560g/m^3/1000 = 1344kgTS/d \\ V &= \frac{4d \cdot 1344kgTS/d}{2.3kgTS/m^3} = 2338m^3 \end{aligned}$$

Assuming that the loss of solids with the effluent is negligible and that the mass of sludge in the secondary settling tanks also is negligible relative to that in the biological reactor the sludge retention time (SRT) can be calculated as follows (Henze *et al.*, 2008, p. 56):

$$SRT = \frac{V}{Q_W} \tag{A.11}$$

In the above equation V represents the volume of the biological reactor and Q_W stand for the waste flow rate. Given the values for SRT and V the waste flow may be derived.

A.2 Modeling organic carbon and nitrogen removal

Model definitions of table A.3 and A.4 are published in Olsson and Newell (1999). In table A.4 stoichiometric terms for oxygen have been left out since in its present implementation oxygen is accounted by a constant value of 5 g/m^3.

Table A.3: Kinetics for carbon and nitrogen removal (Olsson and Newell, 1999, p. 67,71)

#	Process	Components						Kinetics
		S_S	X_S	Ammonium	Nitrate	X_{Het}	X_{Aut}	
1	Aerobic heterotrophic growth	$-\frac{1}{Y_H}$		$-i_{XB}$		1		$\hat{\mu_H}\left(\frac{S_S}{K_S+S_S}\right)\left(\frac{S_O}{K_{OH}+S_O}\right)X_H$
2	Anoxic heterotrophi growth	$-\frac{1}{Y_H}$		$-i_{XB}$	$\frac{Y_H-1}{2.86Y_H}$	1		$\hat{\mu_H}\left(\frac{S_S}{K_S+S_S}\right)\left(\frac{K_{OH}}{K_{OH}+S_O}\right)\left(\frac{S_{NO}}{K_{NO}+S_{NO}}\right)X_H$
3	Aerobic autotrophic growth			$-i_{XB}-\frac{1}{Y_A}$	$\frac{1}{Y_A}$		1	$\hat{\mu_A}\left(\frac{S_{NH}}{K_{NH}-S_{NH}}\right)\left(\frac{S_O}{K_{OA}+S_O}\right)X_A$
4	Decay of heterotrophs		$1-f_P$	$i_{XB}-f_P i_{XP}$		-1		$b_H X_H$
5	Decay of autotrophs		$1-f_P$	$i_{XB}-f_P i_{XP}$			-1	$b_A X_A$
6	Hydrolysis	1	-1					$K_H\left(\frac{X_S/X_H}{K_X+X_S/X_H}\right)X_H$

Table A.4: Kinetics for simple fermentation model (Olsson and Newell, 1999, p. 76)

Process	Components			Kinetics
	insoluble organic carbon	fermentable substrate	fatty acids	
Hydrolysis	-1	$1-f_{si}$		$K_h\frac{(X_S/X_H)}{K_X+(X_S/X_H)}X_H$
Fermentation		-1	1	$q_{FE}\frac{S_F}{K_{FE}+S_F}\frac{S_{ALK}}{K_{ALK}+S_{ALK}}X_H$

Table A.5: Used coefficients according to example 3.8 and 4.1 in Olsson and Newell (1999)

Parameter	Amount	Unit
Y_H	0.63	g/g
Y_A	0.24	g/g
$\hat{\mu_H}$	5.00	$1/d$
$\hat{\mu_A}$	1.00	$1/d$
K_S	20.00	
K_{OH}	0.20	
K_{NO}	0.50	
K_{NH}	1.00	
K_{OA}	0.50	
K_H	3.00	
K_X	0.10	
i_{XB}	0.10	
i_{XP}	0.10	
f_P	0.10	
b_H	0.40	$1/d$
b_A	0.15	$1/d$

B
Propositional Logic and Predicate Logic

B.1 Propositional Logic

A simple type of logic is described by the so called propositional logic[1]. As representation language propositional logic is a *declarative language* since the meaning of propositions is based on the relations between propositions. This fact is also expressed by the compositional character of propositional logic. The meaning of a proposition is derived by the involved components. Furthermore as a logic based representation language propositional logic enables the representation of knowledge independent of the represented context as well as in an unambiguous manner.

Syntax

In propositional logic a single proposition is encoded by a single letter. Such a proposition is referred to as an *atomic formulae*. For example may p represent the proposition: `Every reactor has an inlet`. This is a very simply way to represent a proposition accepting loss of detail. This loss of detail may be an advantage if the internal structure of the proposition is not important but the patterns of implications between them (Sowa, 2000, p. 12 et sqq.). Atomic formulas can be connected by logical connections (syn.: boolean operators) to form *complex formulas* or *well-formed formulas*. The most common five logical connections are listed in table B.1. In order to reflect the precise meaning of a complex formulae the use of brackets may be necessary. In the order of priority the logical connectors are (starting with the highest priority): negation, conjunction, disjunction, implication, equivalence. Due to the different valence of the connectors is the expression $\neg P \vee Q \wedge R \Rightarrow S$ equal in its meaning to the expression $((\neg P) \vee (Q \wedge R)) \Rightarrow S$.

[1] deutsch: *Aussagenlogik* or *Boolesche Logik*

Table B.1: Logical connections (i.e. logical junctions)

Name	**Symbol**	**Intended meaning**
Conjunction	$\cap$ or $\wedge$	and
Disjunction	$\cup$ or $\vee$	or
Negation	$\sim$ or $\neg$	not
Material implication	$\supset$ or $\Rightarrow$	if-then
Equivalence	$\equiv$ or $\Leftrightarrow$	if-and-only-if

Table B.2: Truth-table

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Semantic

In the case of propositional logic semantic defines the rules to proof a formula in the context of a given model. In other words semantics for propositional logic gives meaning to a formula where the truth-values can only be true or false. Following the rules of semantics the truth-value of a complex formula or a model can be reduced to the truth-value of an atomic formula. Those rules are summarized in so called truth-tables. Table B.2 shows the truth table for the latter introduced five logical connectors. The truth-value of each formula or model can be determined by applying such a truth table recursively for each atomic formula. For example may p_1, p_2, p_3 be propositions in the model $m_1 = \{p_1 = \text{false}, p_2 = \text{false}, p_3 = \text{true}\}$. Given the formula $\neg p_1 \wedge (p_2 \vee q_3)$ model m_1 can recursively be solved as follows $\neg\text{true} \wedge (\text{false} \vee \text{true}) = \text{true} \wedge \text{true} = true$.

Inference

In the case of propositional logic there are two ways to proof if a given proposition is true in the context of a given knowledge base. In the first approach all possible models are listed to proof if a particular proposition is true in all models where also the knowledge base is true. The number of possible models in a knowledge base is equal to 2^n with n propositions. The process to perform such a logical inference is referred to as *inference-algorithm* (syn.: model checking). In the second approach so called *inference rules* are used to form a logical

Table B.3: Logical equivalence

Commutativity of $\wedge$	$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$
Commutativity of $\vee$	$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$
Associativity of $\wedge$	$((\alpha \wedge \beta) \vee \gamma) \equiv (\alpha \wedge (\beta \vee \gamma))$
Associativity of $\vee$	$((\alpha \vee \beta) \wedge \gamma) \equiv (\alpha \vee (\beta \wedge \gamma))$
Double negation	$\neg(\neg\alpha) \equiv \alpha$
Contraposition	$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \Rightarrow \neg\beta)$
Implication	$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$
Elimination of biconditionality	$(\alpha \Leftrightarrow \beta) \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
de Morgan	$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$
Distributivity of $\wedge$	$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$
Distributivity of $\vee$	$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

chain between the proposition to proof and the corresponding propositions in the knowledge base. Related to logical inference is the so called logical equivalence. This means if two propositions α and β are true in the same models they are *logical equivalent*, which is denoted by $\alpha \equiv \beta$. Furthermore it can be followed that $\alpha \equiv \beta$ is true if $\alpha \vdash \beta$ and $\beta \vdash \alpha$. Logical equivalence is used to transform terms to enable the use of inference rules. The most important types of logical equivalence are listed in table B.3.

Besides logical equivalence also the concept of validity and satisfiability are required when rules of inference are applied. A term is a *valid term* if the term is true in all models (e.g. $P \vee \neg P$). From this it follows that for all terms α and β $\alpha \vdash \beta$ is true if the term $\alpha \Rightarrow \beta$ is valid. Further more a term α is a *satisfiable term* if there is at least one model for which α is true. It follows there from that $\alpha \vdash \beta$ is true if it can be proofed that $(\alpha \wedge \neg\beta)$ is not satisfiable, which is referred to as *proofed by contradiction*. Following the latter introduced prerequisites the rules of inference can be applied. The most common rules of inference are listed in table B.4. The notation $\frac{\alpha,\beta}{\gamma}$ thereby is read as: given α and β, γ is inferred[2].

B.2 Predicate Logic

A higher expressiveness than by Propositional Logic is offered by Predicate Logic of first order[3]. Predicate Logic is also referred to as First Order Logic (FOL) because these types of logic allow the recognition of objects. In other

[2] can also be written as $\alpha, \beta \vdash \gamma$

[3] deutsch: *PrÃ¤dikatenlogik*

Table B.4: Rules of inference

Modus ponens	$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$
Modus tollens	$\frac{\alpha \Rightarrow \beta, \neg\alpha}{\neg\beta}$
Conjunction	$\frac{\alpha, \beta}{\alpha \wedge \beta}$ $\frac{\alpha \wedge \beta}{\alpha}$ $\frac{\alpha \wedge \beta}{\beta}$
Disjunction syllogism	$\frac{\alpha \wedge \beta, \neg\alpha}{\beta}$ $\frac{\alpha \wedge \beta, \neg\beta}{\alpha}$
Hypothetical syllogism	$\frac{(\alpha \wedge \beta), (\beta \wedge \gamma)}{(\alpha \wedge \gamma)}$
Addition	$\frac{\alpha}{\alpha \vee \beta}$
Subtraction	$\frac{\alpha \wedge \beta}{\alpha}$
Resolution	$\frac{\alpha \wedge \beta, \neg\beta \wedge \gamma}{\alpha \wedge \gamma}$ $\frac{\alpha \Rightarrow \beta, \neg\beta \Rightarrow \gamma}{\alpha \Rightarrow \gamma}$

words objects are granted a first class status in predicate logic. In this sense do types of higher-order logic treat additionally function and relations as objects (Russel and Norvig, 2003, p. 307).

Within propositional logic the proposition `all reactors have an inlet` have been represented by a single symbol such as p. Thereby the inner structure of the proposition was inaccessible. In predicate logic this proposition can be broken down into smaller parts. The part `all reactors` is referred to as *subject* and the second part `have an inlet` is referred to as *predicate*. The predicate thereby is compiled by the *verb* `have` and the *object* `an inlet`.

Syntax

In predicate logic there are logical and non-logical symbols. The first type of logical symbols are *variables*. Throughout this work variables are denoted by small letters such as x, y, z. The second type of logical symbols are *logical connectives* equal to the ones presented in table B.1. Additional to the already defined connectives there are so called *quantifiers*. The symbol $\forall$ is called the *universal quantifier*, the combination $\forall x$ may be read *for every x*. The second quantifier is the *existential quantifier* $\exists$. The combination $\exists x$ may be read *there exists an x such that*. The third type of logical symbols in predicate logic are punctuations and brackets as they have been already introduced in propositional logic. Besides logical symbols there are non-logical symbols which have an application depending meaning. There are two types of such symbols function symbols and predicate symbols. *Function symbols* are expressed throughout this work in uncapitalised mixed case, e.g. *hasInlet* or more generally using

a, b, c, f, g, and h. *Predicate symbols* are expressed throughout this work in capitalised mixed case, e.g. *LargerThan* or more generally using P, Q, R. Function and predicate symbols can be more differentiated by introducing the definition of arity. *Arity* is a non-negative integer stating how many argument are included within a function or predicate symbol. Function symbols with an arity equal to zero are referred to as *constants*. Predicate symbols of arity 0 are sometimes referred to as *propositional symbols*. In general it can be stated that constants, predicates symbols and functional symbols represent objects, relations and functions respectively. The syntax of predicate logic allows two legal expressions *terms* and *formulas*. Any term (as smallest unit) must satisfy at least the following conditions:

- every variable is a term;
- if $t_1, ...t_n$ are terms, and f is a function symbol of arity n, then $f(t_1, ...t_n)$ is a term.

A set of formulas in predicate logic must at least satisfy the following conditions:

- if $t_1, ...t_n$ are terms, and P is a predicate symbol of arity n, then $P(t_1, ...t_n)$ is a formula;
- if t_1 and t_2 are terms, then $t_1 = t_2$ is a formula;
- if α and β are formulas, and x is a variable, then $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $\forall x.\alpha$ and $\exists x.\alpha$ are formulas.

Formulas of the first two types (containing no other simpler formulas) are referred to as *atomic formulas* or *atoms*. A *sentence* in predicate logic is any formula without free variables. Therefore sentences in predicate logic are used to represent knowledge.

Semantics

It is not possible by the semantic of predicate logic to tell an observer what a constant reflecting a real world object is, or what has been meant exactly by the designer. Instead the meaning of a sentence in predicate logic is a function

Table B.5: Logical equivalence

$\neg(\forall x)F(x) \equiv$	$(\exists x)\neg F(x)$
$\neg(\exists x)F(x) \equiv$	$(\forall x)\neg F(x)$
$(\forall x)(\forall y)F(x) \equiv$	$(\forall y)(\forall x)F(x,y)$
$(\exists x)(\exists y)F(x) \equiv$	$(\exists y)(\exists x)F(x,y)$
$(\forall x)(F(x) \wedge G(x)) \equiv$	$(\forall x)F(x) \wedge (\forall x)G(x)$
$(\exists x)(F(x) \vee G(x)) \equiv$	$(\exists x)F(x) \vee (\exists x)G(x)$

of the interpretation of the predicate and function symbols. The relation between reality and what can be interpreted from predicate logic is subsumed by the following views on reality. First, there are objects in the world. Second, for any predicate P of arity 1, some of the objects will satisfy P and some will not. An interpretation of P settles the question, deciding for each object whether it has or does not have the property in question. Beyond this no other aspects of the world matter. The *model theoretical semantics* therefore defines the foundation of interpretation which is the theory T of concern or more particular the knowledge base, KB. The concerned theory thereby holds all formulas as basis for interpretation. It follows further that an interpretation I is a model for T if $I \vdash G$ is true for any formula G in T. It can now be stated that a formula F is a *logical consequence* from T if a model of T is also a model of F. This is represented by $T \vdash F$ meaning that formula F is a logical consequence from T. To be able to derive new formulas out of existing ones it is necessary to remodel (syn.: transform) existing formulas. An essential concept therefore is again *logical equivalence* as it has been introduced in the latter section. Two formulas F and G are logical equivalent (i.e. semantically equivalent)if both $F \vdash G$ and $G \vdash F$ are true. Logical equivalence between F and G is represented by $F \equiv G$. Some examples of types of equivalence regarding predicate logic are listed in table B.5.

For any formula there exists an unlimited number of equivalent formulas (e.g. F can be expressed by $\neg\neg F$ and so on) (Brachman and Levesque, 2004, p. 49 ff.). To be able to perform automatically logical inference it is therefore necessary to transform any formula of concern into a normalized form (i.e. equivalence class). One type of such a normalized form is the so called Conjunctive Normal Form (CNF) or *clausal form* (Russel and Norvig, 2003, p. 273). A for-

mula is in CNF if it is a conjunction of clauses, where a clause is a disjunction of literals. By the following four steps a given formula can be remodeled into the CNF (Russel and Norvig, 2003, p. 368):

Negation normal form Eliminate implications such that $x \rightarrow y$ to $\neg x \vee y$. Move $\neg$ inwards so that it appears only in front of an atom (e.g. $\neg \forall x p \equiv \exists x \neg p$). As result all implications and equivalences are eliminated and negations are attached directly to atoms (Brachman and Levesque, 2004, p. 50).

Prenex normal form Move all quantifiers to the beginning. Therefore variables must be standardized. This means if a variable is used more than once a second variable is introduced. For example is the expression $(\forall x P(x)) \vee (\exists x Q(x))$ replaced by the expression $(\forall x P(x)) \vee (\exists y Q(y))$. Now the quantifiers can be moved to the beginning of the expression such that: $(\forall x)(\exists y)(P(x) \vee Q(y))$.

Skolemized prenex normal form Eliminate all existential quantifiers (Brachman and Levesque, 2004, p. 64). Thereby an expression such as $\exists x P(x)$ is resolved to $P(A)$ where A is a new constant (i.e. *skolem constant*). Thereby a skolem constant can be introduced if no universal quantifier is left of the existential quantifier to be eliminated. In the case that there is a universal quantifier left of the existential quantifier to be removed a so called *skolem function* is used to replace the existential quantifier. After all existential quantifiers have been replaced all universal quantifiers can be eliminated.

CNF Transform by logical equivalence the remaining formula as conjunctions of disjunctions (e.g. $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$).

After two formulas have been resolved into CNF it is much easier to compare the two formulas according to logical equivalence since the are syntactically equivalent. Through the process of resolution a violation regarding logical equivalence has been introduced within the step of skolemization. For a given formula F and its CNF K it is $F \not\equiv K$. Nevertheless if F is unsatisfiable also K is unsatisfiable (Brachman and Levesque, 2004, p. 68).

Inference

As stated in the context of propositional logic the objective of logical inference is to proof that a formula F_0 is a logical consequence of a given knowledge base KB ($KB \vdash F_0$) where KB is a collection of sentences $F_1, ..., F_n$. A particular proof system called *resolution* attempts to solve this objective through deducing contradictions (Russel and Norvig 2003, p. 368 and Brachman and Levesque 2004, p. 49 ff.). Starting with the attempt to proof that $\{F_1, ..., F_n\} \vdash F_0$ which can be transformed to $F_1 \wedge ... \wedge F_n \rightarrow F_0$ through resolution it is attempted to proof that $\neg(F_1 \wedge ... \wedge F_n \rightarrow F_0)$ is unsatisfiable. Therefore all formulas are transferred into CNF and the resulting negated collection of sentences $G_1 \wedge ... \wedge G_k$ is used to deduct a contradiction.
The inference rule referred to as *binary resolution* can be applied in an entailment procedure shown below. To proof a contradiction within a set of clauses M for clauses represented in propositional logic the following steps must be applied:

1. Resolve two clauses out of set M to form a new clause K. Is the new generated clause K solely composed by a single atom (or negated atom) it is referred to as an empty clause (represented by $\perp$).
   For examples the two clauses $(p \wedge q)$ and $(r \wedge \neg q)$ are resolved in the new clause $(p \wedge q) \vee (r \wedge \neg q)$.

2. If $K = \perp$ a contradiction is detected.

3. If $K \neq \perp$, add K to M and start with step 1.

To apply binary resolution for knowledge bases in predicate logic additional relations must be considered by help of substitutions. More detailed descriptions are provided by Brachman and Levesque (2004, p. 55 ff.) and Russel and Norvig (2003, p. 370).
Concluding the following properties about predicate logic can be summarized. Knowledge bases represented in predicate logic are monotonic (Brachman and Levesque, 2004, p. 209). This means, if new facts are added to an existing knowledge base no entailments deductible from the previous KB status get lost. Knowledge Bases in predicate logic are semi-decidable. This states that

all true entailments are traceable. In contrast not all false consequence can be proofed. Depending on how strong the use of quantifiers is restricted various subsets of predicate logic emerge. Some of them are: propositional logic, horn-clauses (Raffeiner, 2005, p. 91), Datalog, and Description Logics.

C
GUI flowsheet finder

The models and algorithms that have been developed within this work for the purpose of flowsheet generation have been implemented into a Java base software tool. This tool may be applied as command line tool or by a GUI. The handling of the GUI is described within this section.
The general workflow as processed by the flow sheet finder goes through the sequential steps of setting up the initial plan, fractionation of initial states and searching for feasible complete plans (see figure 4.1). At present the use can only select predefined initial plans which have been introduced as use cases in section 3.3. Any further development of the flow sheet finder would required the implementation of a wizard to define user specific initial plans.
The GUI panel is shown in figure C.1. The panel is divided into four areas. In area (1.1) the initial plan is vitalized as graph. As soon complete plans have been identified they can be shown within this area. Any unit selected in area (1.1) is shown in detail in area (1.2). The provided information cover preconditions, properties and the in and outgoing states of a selected unit. In section (1.3) the ranked list of identified plans is listed. Area (1.4) served as console to provide information of an ongoing search or other relevant information.
Figure C.2 shows the GUI after a completed search, where only one complete plans has been found. To take influence on the search a preference panel (figure C.3) allows the adjustment of applied search parameter as described in section 4.8.5. Eventually identified plans can be exported in DOT format and as table, see figure C.4.

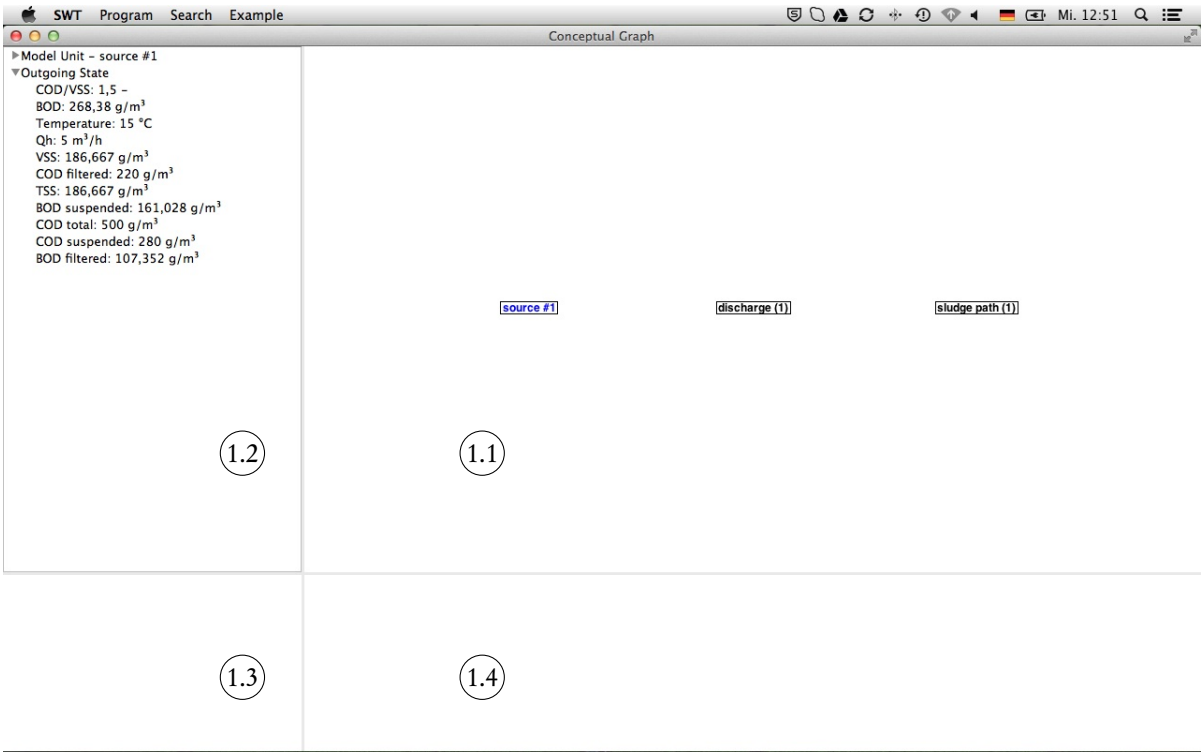


Figure C.1: GUI in setup mode, showing initial plan

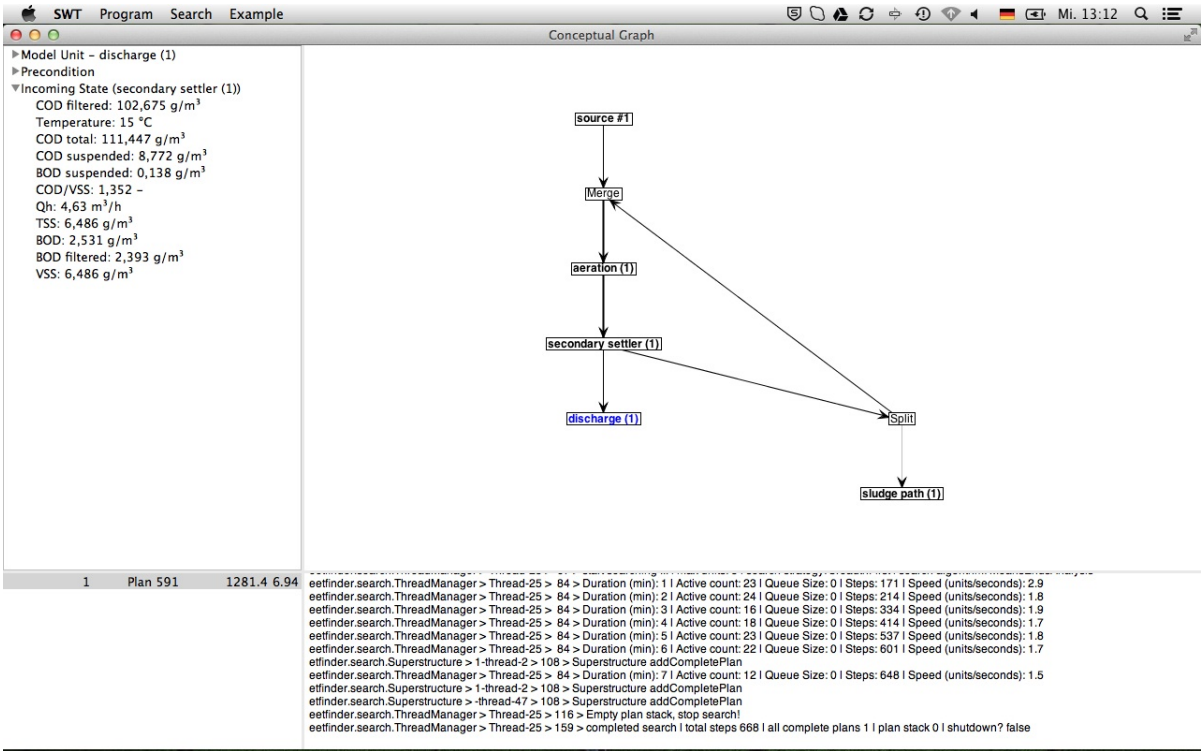


Figure C.2: GUI showing search results

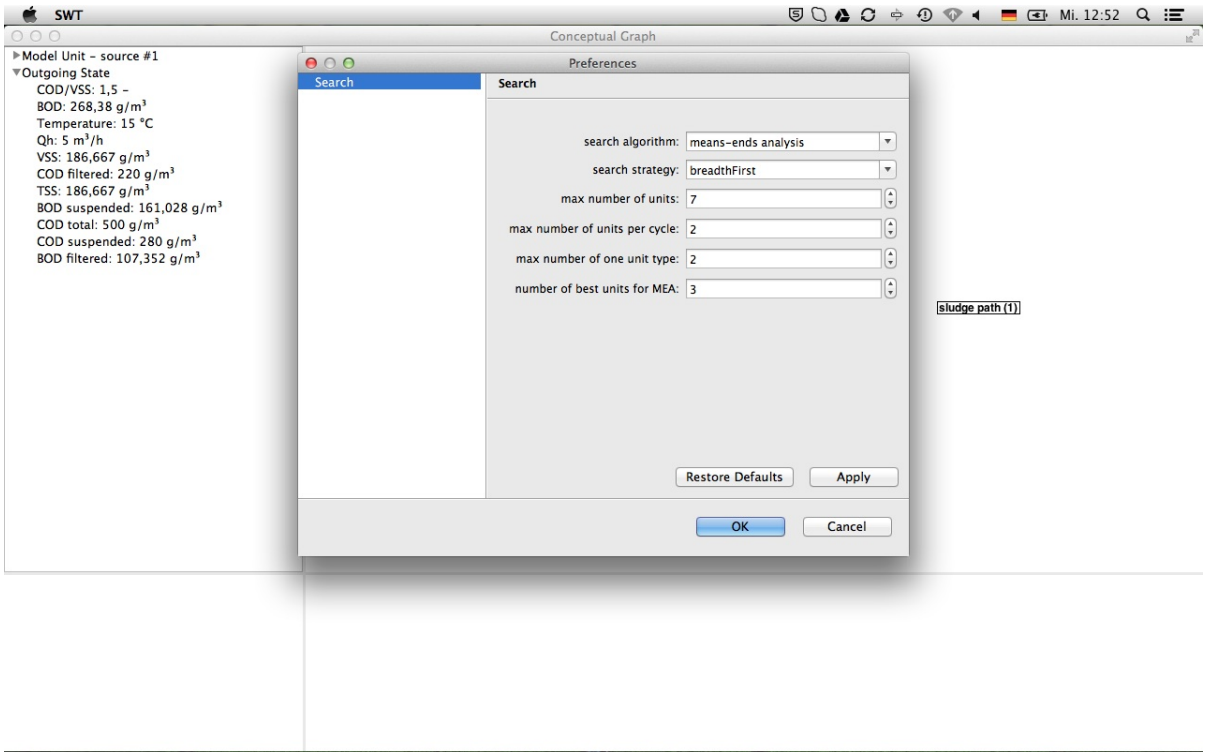


Figure C.3: Preference panel

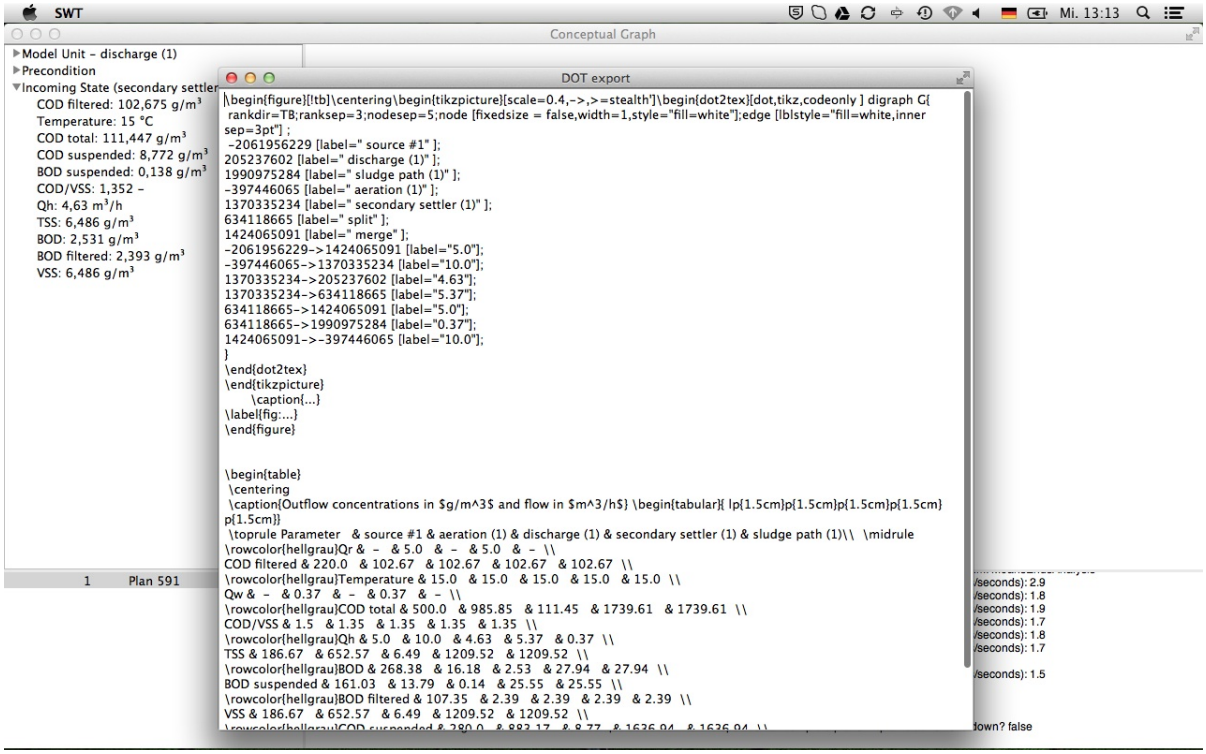


Figure C.4: Export window for identified flowsheets

Bisher sind in der Schriftenreihe des Instituts für Siedlungs- und Industriewasserwirtschaft der TU Dresden folgende Bände erschienen:

Band	Titel	
1	**Theorie und Praxis der Industriewasserwirtschaft - Beitrag zum Umweltschutz** Kolloquium, Dresden 1988	*vergriffen*
2	**Aktuelle Fragen zur Abwasser- und Schlammbehandlung sowie zum Stoffrecycling aus Abwässern** Kolloquium, Dresden, 1990	*vergriffen*
3	**Theorie und Praxis der Industriewasserwirtschaft** Kolloquium, Dresden, 1991	
4	**Aktuelle Fragen der Wasserversorgung und Abwasserbehandlung** Kolloquium anläßlich des 65. Geburtstags von Prof. Dr. Ing. habil. H. Kittner und des 60. Geburtstages von Prof. Dr. Ing. habil. J. Hackenberger, Dresden, 1991	
5	**Optimierung des Elektrodialyse-Verfahrens in Auswertung von Langzeituntersuchungen (1995)** Danz, K. (Diss.)	
6	**Beitrag zur Steuerungsoptimierung von Wasserverteilungssystemen (1992)** Geisendörfer, M. (Diss.)	*vergriffen*
7	**Wahrscheinlichkeitsanalyse von Hochwasserdurchflüssen (1996)** Kluge, C. (Diss.)	*vergriffen*
8	**Möglichkeiten und Grenzen des Ozoneinsatzes zur Minimierung der Trihalogenmethanbildung nach einer Chlorung (1996)** Heiser, H. (Diss.)	
9	**Aktuelle Fragen der Wasserversorgung und Abwasserbehandlung** Ehrenkolloquium anläßlich des 65. Geburtstags von Prof. Dr. Ing. habil. J. Hackenberger, Dresden, 1997	
10	**Aktuelle Fragen der Wasserversorgung und Abwasserbehandlung** Ehrenkolloquium anläßlich des 75. Geburtstags von Doz. i.R. Dr. Ing. habil. J. Gruhler, Dresden, 1997	*vergriffen*
11	**Naturnahe und technische Klein- und Kleine Kläranlagen im Vergleich** 11. Kolloquium, Dresden, 1997	*vergriffen*
12	**Naturnahe und technische Klein- und Kleine Kläranlagen im Vergleich** 12. Kolloquium, Dresden, 1998	*vergriffen*
13	**Die Trinkwasserversorgung auf dem Weg ins neue Jahrtausend** Kolloquium, Dresden, 1999	*vergriffen*
14	**Erfahrungen und neue Entwicklungen in der Abwasserbehandlung** Kolloquium, Dresden, 1999	

Band	Titel
15	**Die Einordnung der Stickstoffrückbelastung aus der anaeroben Schlammstabilisierung in den Bilanzrahmen einer kommunalen Abwasserreinigungsanlage (2000)** Kühn, V. (Diss.)
16	**Stoffhaushalt in der Siedlungsentwässerung (2000)** Tagungsband, Dresdner Kolloquium zur Siedlungswasserwirtschaft
17	**Aktuelle Probleme der Abwasserbehandlung (2001)** Tagungsband, Dresdner Kolloquium zur Siedlungswasserwirtschaft
18	**Thermophile Vergärung von Mischsubstraten (2001)** Dornack, Ch. (Diss.)
19	**Innovationen in der Abwasserableitung und Abwassersteuerung (2002)** Tagungsband, Dresdner Seminar Wasserbau und Wasserwirtschaft
20	**Bemessung und Betrieb von Anlagen zur Grundwasseraufbereitung (2002)** Wingrich, H.
21	**Ein Beitrag zur Bilanzierung von Bodenfiltern (2002)** Müller, V. (Diss.)
22	**Wasserversorgung 2002/Probleme - Entwicklungen - Anwendungen (2002)** Tagungsband, Dresdner Kolloquium zur Siedlungswasserwirtschaft
23	**Abwasserseminar 2003 mit Verabschiedung von Herrn Prof. Dr.-Ing. habil. Klaus Lützner (2003)** Tagungsband, Dresdner Kolloquium zur Siedlungswasserwirtschaft
24	**Absetzverfahren in der Abwasserreinigung (2004)** Tagungsband, Dresdner Kolloquium zur Siedlungswasserwirtschaft
25	**Hydrologische Modellierung urbaner Nährstoffeinträge in Gewässer auf Flussgebietsebene (2006)** Biegel, M. (Diss.)
26	**Beitrag zur Anwendung hydraulischer Rohrnetzmodelle für die Wassergütemodellierung (2006)** Beilke, G. (Diss.)
27	**Assessment of sewer leakage by means of exfiltration measurements and modelling tests (2007)** Rutsch, M. (Diss.)
28	**Spatial classification methods for efficient infiltration measurements and transfer of measuring results (2007)** Franz, T. (Diss.)
29	**Integration in der Abwasserentsorgung (2007)** Tagungsband, Dresdner Kolloquium zur Siedlungswasserwirtschaft
30	**Prozesswasserbehandlung - Problemstellungen und Lösungen (2008)** Tagungsband, Dresdner Kolloquium zur Industriewasserwirtschaft

Band	Titel
31	**Zur Behandlung und Verwertung von Rückständen aus der Oberflächenwasseraufbereitung (2009)** Reißmann, F. G. (Diss.)
32	**Interaktionen bei der Modellierung von Stofftransport, Sedimenthaushalt und Abfluss in der Siedlungsentwässerung (2009)** Gebhard, V. (Diss.)
33	**Anpassung der Abwassersysteme an veränderte Randbedingungen (2010)** Tagungsband, Dresdner Kolloquium zur Industriewasserwirtschaft
34	**Erschließung von Biogaspotenzialen aus überschussschlamm mit Hilfe der Kombination aus Desintegration und anaerober Schlammstabilisierung (2012)** Barth, M. (Diss.)
35	**Modellierung der Interaktion zwischen Grundwasser und Kanalisation (2012)** Karpf, C. (Diss.)
36	**Studying the contribution of urban areas to fine sediment and associated element contents in a river bed (2013)** David, T. (Diss.)
37	**A conceptual framework to characterise the impacts of urban wastewater systems on receiving water quality (2013)** Blumensaat, F. (Diss.)
38	**Auswirkungen des demographischen Wandels auf den Betrieb zentraler Abwassersysteme (2013)** Tränckner, J. (Habil.)

Diese Bände sind zu beziehen über:

Institut für Siedlungs- und Industriewasserwirtschaft
Technische Universität Dresden
D - 01062 Dresden
Tel. (0351)463-32337, Fax (0351)463-37204, e-mail: isi@mail.zih.tu-dresden.de

Preise: Bände 1 bis 15: 15,00 EUR
ab Band 16: 20,00 EUR

Mitglieder des Fördervereins erhalten die Bände kostenlos.