

Schriftenreihe

Band 154

Andreas Gebhard

**Creation of an Information Management
System for Tribology Laboratories
and its Application to Transfer Film
Luminance Analysis**

Prof. Dr.-Ing. Ulf Breuer (Hrsg.)

IVW - Schriftenreihe Band 154

Leibniz-Institut für Verbundwerkstoffe GmbH

Kaiserslautern

Andreas Gebhard

**Creation of an Information Management
System for Tribology Laboratories
and its Application to Transfer Film
Luminance Analysis**

Bibliografische Information Der Deutschen Bibliothek

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

Bibliographic information published by Die Deutsche Bibliothek

Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <<http://dnb.dnb.de>>.

Herausgeber: Leibniz-Institut für Verbundwerkstoffe GmbH
Prof. Dr.-Ing. Ulf Breuer
Erwin-Schrödinger-Straße 58
Technische Universität Kaiserslautern
67663 Kaiserslautern
<http://www.ivw.uni-kl.de>

Verlag: Leibniz-Institut für Verbundwerkstoffe GmbH

Druck: pri-me Printservice & Medienservice
Barbarossastraße 1
67655 Kaiserslautern
D-386

© Leibniz-Institut für Verbundwerkstoffe GmbH, Kaiserslautern 2023

Alle Rechte vorbehalten, auch das des auszugsweisen Nachdrucks, der auszugsweisen oder vollständigen Wiedergabe (Photographie, Mikroskopie), der Speicherung in Datenverarbeitungsanlagen und das der Übersetzung.

Als Manuskript gedruckt. Printed in Germany.

ISSN 1615-021X

ISBN: 978-3-944440-51-4

Creation of an Information Management System for Tribology Laboratories and its Application to Transfer Film Luminance Analysis

Vom Fachbereich Maschinenbau und Verfahrenstechnik
der Technischen Universität Kaiserslautern
zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte

Dissertation

von

Herrn

Dipl.-Chem. Andreas Gebhard
aus Ludwigshafen am Rhein

Tag der mündlichen Prüfung: 24.10.2022

Dekan

Prof. Dr.-Ing. Tilmann Beck

Prüfungsvorsitzender

Prof. Dr.-Ing. Joachim Hausmann

1. Berichterstatter

Prof. Dr.-Ing. Ulf Breuer

2. Berichterstatter

Prof. Dr.-Ing. Stefan Deßloch

D386

Acknowledgements

First, I would like to thank Prof. Ulf Breuer for supervising this thesis, Prof. Stefan Deßloch for reviewing it and Prof. Joachim Hausmann for chairing the examination committee. Additional thanks go to Dr.-Ing. Bai-Cheng Jim for inventing Transfer Film Luminance Analysis with me and to all members of the competence field tri-bology for their fruitful collaboration. Special thanks go to Dr.-Ing. Bernd Wetzel for his continued personal and academic support as well as to Stefan Brunner and Markus Hentzel for their help in the practical implementation of many key concepts of this work.

Kaiserslautern, November 2022

Contents

Abstract.....	iv
Kurzfassung.....	vi
Glossary.....	ix
1 Laboratory Information Management in Tribology.....	1
1.1 State of the Art.....	1
1.2 Shortcomings of the state of the art, objectives of this work and solution procedure.....	8
2 Fundamentals.....	11
2.1 Entity-Relationship Data Modeling.....	11
2.2 Object oriented programming.....	13
2.2.1 Functional programming.....	13
2.2.2 Object definition.....	14
2.2.3 Inheritance, Generalization and Specialization.....	16
2.2.4 Extension and method overriding.....	17
2.2.5 Association, Aggregation, Property Propagation and Duck Typing.....	18
2.2.6 Composition.....	19
2.2.7 Uniform Access Principle.....	19
2.3 Relational Database Management Systems.....	19
2.3.1 Identity.....	20
2.3.2 Relationships.....	21
2.3.3 Data Quality and Data Integrity.....	21
2.4 Object relational mapping.....	23
2.5 Representation of specialization in RDMS.....	23
2.6 Web applications.....	24
3 Requirements for a LIMS for tribology laboratories.....	25
3.1 Workflow analysis.....	25
3.2 Summary of requirements.....	29
4 Sample implementation: <i>Atlas</i>	31
4.1 Components.....	31
4.2 Hardware, operating system and parallel computation strategy.....	32
4.3 Primary keys, identity and labelling system.....	32
4.4 Materials and material lots database.....	33
4.5 Test specimens.....	34
4.5.1 Domain.....	34
4.5.2 Data modeling.....	36

4.5.3	Atypical inverse property propagation	38
4.5.4	Test specimen labelling and archiving.....	42
4.5.5	GUI elements for workflow support	44
4.6	Tribometers.....	45
4.6.1	Domain	45
4.6.2	Data model	45
4.7	Wear Tests	46
4.7.1	Domain	46
4.7.2	Data model	48
4.7.3	Attributes and Associations.....	51
4.7.4	Creation of wear tests	55
4.8	Test setup and execution of wear tests	59
4.9	Measurement data files	63
4.9.1	Domain	63
4.9.2	Uniform data format and naming convention	64
4.9.3	Type specific domain requirements.....	70
4.9.4	Data model	71
4.9.5	Server-side storage.....	72
4.9.6	Data file upload and automatic standardization	74
4.9.7	Automatic evaluation and result persistence	74
4.10	Visualization of results	79
4.11	Assessment and storage of the validity of results.....	87
4.12	Automatic report generation	88
4.13	Limitations and known issues.....	89
5	Transfer Film Luminance Analysis	91
5.1	State of the Art.....	91
5.2	Shortcomings of the state of the art, objectives and solution procedure	97
5.3	Photo-optical transfer film luminance analysis (TLA)	101
5.3.1	Preliminary considerations and assumptions	101
5.3.2	Measurement principle, test setup and data recording	101
5.3.3	Image processing and data extraction.....	105
5.3.4	Data Processing, Aggregation and Plotting	109
5.3.5	Polishing Detection	111
5.3.6	Limitations and known issues.....	118
5.4	Automation and integration into <i>Atlas</i>	120
6	Summary.....	125
	References	129

Figures.....	138
Tables.....	143
Code listings	144
Appendix.....	146
Publications and conference contributions	149
Supervised Theses	152

Abstract

In tribology laboratories, the management of material samples and test specimens, the planning and execution of experiments, the evaluation of test data and the long-term storage of results are critical processes. However, despite their criticality, they are carried out manually and typically at a low level of computerization and standardization. Therefore, formats for primary data and aggregated results are wildly different between laboratories, and the interoperability of research data is low. Even within laboratories, low levels of standardization, in combination with ambiguous or non-unique identifiers for data files, test specimens and analysis results greatly reduce data integrity and quality. As a consequence, productivity is low, error rates are high, and the lack or low quality of metadata causes the value of produced data to deteriorate very quickly, which makes the re-use of data, e.g. for data mining and meta studies, practically impossible.

In other fields of science, these are mitigated by the use of Laboratory Information Management Systems (LIMS). However, at the moment, such systems do not exist in tribological research. The main challenge for the implementation of such a system is that it requires extensive interdisciplinary knowledge from otherwise very disparate fields: tribology, data and process modelling, quality management, databases and programming. So far, existing solutions are either proprietary, very limited in their scope or focused on merely storing aggregated results without any support for laboratory operations.

Therefore, this thesis describes fundamentals of information technology, data modelling and programming that are required to build a LIMS for tribology laboratories. Based on an analysis of a typical workflow of a tribology laboratory, a data model for all relevant entities and processes is designed using object-relational data modelling and object-oriented programming and a relational database is used to provide a reference implementation of such a LIMS. It provides critical functionalities like a materials database, test specimen management, the planning, execution and evaluation of friction and wear tests, automated procedures for tribometer parameterization and data transmission, storage and evaluation and for aggregating individual tests into test sets and projects. It improves the quality and long-term usability of data by replacing error-prone human processes by automated variants, e.g. automated collection of metadata and data file transmission, homogenization

and storage. The usefulness of the developed LIMS is demonstrated by applying it to Transfer Film Luminance Analysis (TLA), which is a newly developed advanced method for the analysis of the formation and stability of transfer films and their impact on friction and wear, but which produces so much data and requires such a large amount of metadata during evaluation that it can only be performed safely, quickly and reliably by integration into the presented LIMS.

Kurzfassung

In tribologischen Laboren sind die Verwaltung von Materialproben und Prüfkörpern, die Planung und Durchführung von Experimenten, die Auswertung von Versuchsdaten und die Langzeitspeicherung von Ergebnissen kritische Prozesse. Ihre Durchführung erfolgt bis heute ganz überwiegend manuell und ist entsprechend fehleranfällig und wenig interoperabel. Eine besondere Herausforderung ist dabei die Erfassung und Speicherung von Metadaten und von semantischen Zusammenhängen zwischen den vielen verschiedenen Entitäten. Während vor allem im Gesundheitswesen Labor-Informationen-Management-Systeme (LIMS) eingesetzt werden, um die Durchführung von Prozessen zu unterstützen, sind solche Systeme in der tribologischen Forschung nur wenig verbreitet. Bestehende Lösungen und aktuelle Bestrebungen sind hauptsächlich reine Ergebnisdatenbanken, die als nachgelagerte Datenspeicher für Labore dienen und daher im eigentlichen Laborbetrieb praktisch keine Rolle spielen. Im Bereich der Prüfkörperverwaltung fehlt es an Informationssystemen, die kritische Laborprozesse wie die Vergabe von Identifikatoren für Werkstoffe und Prüfkörper, das Speichern von Prüfkörper-Merkmalen auf unterschiedlichen Aggregationsebenen (z.B. einzeln oder als Charge) oder die systematische Archivierung von Probekörpern unterstützen. Im Bereich der Versuchserstellung und Verwaltung gibt es aktuell kein einheitliches Informationssystem für die enorme Vielfalt von Versuchsarten, Prüfkörperanordnungen, Bewegungsarten und Versuchsabläufen. Im Bereich der Versuchsdurchführung bestehen die Hauptherausforderungen im Fehlen automatisierter Prozesse für die Parametrierung von Tribometern und der Verwaltung der primären Versuchsdaten. Entsprechend gering ist der Computerisierungsgrad bei Auswertung und Darstellung von Messdaten sowie bei deren Aggregation. Bestehende Lösungen sind deshalb häufig auf einzelne Arbeitsgruppen oder gar nur einzelne Personen beschränkt.

Diese Arbeit befasst sich daher mit dem Entwurf und der Implementierung eines exemplarischen Informationsmanagement-Systems für Tribologielabore, mit dem Ziel die genannten Lücken zu schließen. Die Basis hierfür ist eine ausführliche Analyse und Beschreibung der Konzepte, der physischen Entitäten und der Prozess in einem typischen Arbeitsablauf eines auf Versuche der Kategorien VI („Mo-

dellversuch mit einfachen Probekörpern“) bis IV („Versuch mit unverändertem Bauteil oder verkleinertem Aggregat“) spezialisierten Tribolabors. Zu den daraus abgeleiteten Anforderungen gehören insbesondere die Unterstützung des Werkstoff- und Prüfkörpermanagements, die Erstellung und Verwaltung tribologischer Versuche, deren Aggregation zu Versuchsreihen und Projekten, die Automatisierung von der Parametrierung von Tribometern, der Übertragung, Speicherung, Auftragung und Auswertung von Messdaten sowie die Projektverwaltung und Berichtserstellung. Zur Erfüllung dieser Anforderungen wird mit Hilfe der Objektrelationalen Datenmodellierung ein ausführliches logisches Datenmodell aller relevanten Entitäten erstellt, inklusive deren semantischer Relationen, und dieses dann mit Hilfe der objektorientierten Programmierung implementiert.

Für die Herausforderung des variablen Detaillierungsgrads bei der Identifikation und Beschreibung von Prüfkörpern (einzeln, als Charge oder als Klasse) wird hierbei ein Modell mit drei assoziierten Klassen entwickelt, die diese unterschiedlichen Aggregationsstufen repräsentieren. Die in einem Tribolabor verwendeten Arten von Prüfkörpern (Ring, Block, Scheibe usw.) werden dann mittels Spezialisierung einer allgemeinen Prüfkörperklasse dargestellt, und zwar auf jeder der drei Aggregationsebenen. Mittels atypischer Eigenschaftspropagierung erlaubt es dieses Modell, Attribute von Prüfkörpern auf jeder der drei Aggregationsebenen zu definieren, und zwar sowohl attributs- als auch prüfkörperspezifisch.

Für tribologische Versuche wird – ebenfalls mittels Spezialisierung, die auf Programmcode-Ebene hauptsächlich mittels Vererbung dargestellt wird – erstmalig ein Datenmodell entwickelt, das die große Vielfalt, die sich aus der unterschiedlichen Anzahl und Geometrie der beteiligten Prüfkörper, der variablen Versuchssegmentierung (Mehrstufenversuche) und der unterschiedlichen Arten und Versuchssegmenten (Gleiten, Stehen, Losbrechen, Oszillieren usw.) ergibt, einheitlich beschreibt. Diese Vereinheitlichung bildet die Grundlage für das vorgestellte allgemeine Verfahren zur automatisierten Parametrierung von Tribometern. Hierfür wird serverseitig eine maschinenlesbare Darstellung eines Versuchs erstellt und an einen Prüfstand übermittelt. Dieser interpretiert die erhaltenen Daten und konfiguriert den Versuchsablauf entsprechend. Durch die Automatisierung dieses bisher fast ausschließlich manuellen Prozesses wird die Häufigkeit von Fehlern in der Versuchsdurchführung stark reduziert.

Als natürliche Verlängerung dieser Maschine-zu-Maschine-Kommunikation wird die automatisierte Übermittlung der von einem Tribometer erstellten Messwertdatei(en), sowie die serverseitige automatisierte Speicherung unter Aufrechterhaltung der logischen Relation zwischen Messwertdatei(en), Versuch, Werkstoff und Projekt beschrieben. Die Grundlage hierfür bildet der Ersatz von benutzerdefinierten Identifikatoren für alle abgebildeten Entitäten durch die Vergabe von maschinenlesbaren Identifikatoren durch das System. Für das spezielle Problem der geringen Standardisierung von Messwertdateien und der daraus resultierenden Vielseitigkeit der zu verarbeitenden Dateiformate wird ein einheitliches Dateiformat vorgeschlagen. Die Umformatierung erfolgt hierbei mittels einer Spezialisierung von Importfiltern bis auf die Ebene der Instanzen des Tribometer-Modells. Hierfür wird erstmalig eine Lösung auf der Basis von dynamisch generierten Instanzmethoden vorgestellt, deren Programmcode die instanzspezifische Domänenlogik enthält und der nicht im Quellcode des Systems, sondern im Datenbankeintrag der jeweiligen Tribometerinstanz gespeichert ist. Auf der Basis dieser Harmonisierung der Messwertdateien werden Lösungen für die automatisierte und vereinheitlichte grafische Auftragung und für die Auswertung von Messwertdateien vorgestellt. Neben der Lösung zahlreicher „Bestandsprobleme“ des Informationsmanagements in Tribolaboren wird der Einsatz des Systems bei der praktischen Durchführung der Transferfilm-Luminanzanalyse demonstriert. Hierbei handelt es sich um ein im Rahmen dieser Arbeit ebenfalls neu entwickeltes Verfahren zur photo-optischen Quantifizierung von Transferfilmen, die sich vor allem im trockenen Gleiten von Kunststoffen und Kunststoffverbundwerkstoffen auf Stahl bilden. Anders als alle bisher in diesem Bereich bestehenden Verfahren ist dieses gleichzeitig quantitativ, zeitauflösend, vollflächig, materialunspezifisch und benötigt keine besonderen Versuchsumgebungen (z.B. Vakuum). Bei seiner Durchführung müssen jedoch nicht nur sehr große Mengen an Bilddaten bearbeitet werden, sondern auch Messdaten aus unterschiedlichen Quellen zusammengeführt und mit einer Vielzahl an Metadaten kombiniert werden. Manuell ist dieses Verfahren aufgrund seiner Komplexität nicht durchführbar. Durch die Integration in das neu entwickelte Laborinformationssystem gelingt sie jedoch sicher, schnell und - aufgrund der Speicherung aller zur Auswertung benötigten Primär- und Metadaten - auch langfristig zuverlässig nachvollziehbar.

Glossary

Abbreviations and Acronyms

1:1	One-to-one (relationship)
1:n	One-to-many (relationship)
AAAA	Authentication, Authorization, Auditing, Accountability
ACTIS	A Computerized Tribology Information System
AFM	Atomic Force Microscopy
AISI	American Iron and Steel Institute
AMD	Advanced Micro Devices
API	Application Programming Interface
ARMES	Abrasion resistant materials expert system, name of a tribological expert system developed in Australia
ASTM	ASTM International, formerly: American Society for Testing and Materials
<i>ATLAS</i>	Automatisches Tribologisches Auswertungssystem, name of the reference implementation for a LIMS for tribology laboratories described in this work
AUS	Australia
BAM	Bundesanstalt für Materialforschung
BeO	Beryllium Oxide
BSD	Berkeley Software Distribution, a family of free software licenses
CGI	Common Gateway Interface
CHN	People's Republic of China
CMDN	China Modern Design Net
CNC	Computerized Numerical Control
COF	Coefficient of Friction
CPU	Central Processing Unit
CRuby	Interpreter for the Ruby Programming Language
CRUD	Create, Read, Update, Delete
CTDB	China Tribology Database, name of a tribological database, expert system and directory for research units and tribologists developed in the People's Republic of China

DDR4	Double Data Rate 4
DLC	Diamond Like Coating
DSL	Domain Specific Language
EDS	Energy Dispersive Spectrometry
EDX	Energy Dispersive X-ray
EN Ax	European Norm for Cast (EN AC) and Wrought (EN AW) Aluminum
ER	Entity-Relationship
ERD	Entity-Relationship-Diagram
FIB	Focused Ion Beam
FP	Functional Programming
FRA	France
FTIR	Fourier-Transform Infrared (Spectroscopy)
GB	Gigabyte
GeO ₂	Germanium Oxide
GER	Federal Republic of Germany
GGr15	Chinese Standard for Bearing Steel
GIL	Global Interpreter Lock
GPL	GNU General Public License
HH:MM:SS	Time format (hours:minutes:seconds)
HRc	Rockwell Hardness, Scale C
HTML	Hypertext Markup Language
I/O	Input/Output
i-TRIBOMAT	Name of a research project in the field of tribo-informatics
IVW	Leibniz-Institut für Verbundwerkstoffe GmbH
KEY	Keyword
LED	Light Emitting Diode
LI(M)S	Laboratory Information (Management) System
MIT	Massachusetts Institute of Technology
Mo-S-Pb	Molybdenum-Sulfur-Lead
MoS ₂	Molybdenum disulfide
MRI	Matz's Ruby Interpreter
MTI	Multi-Table Inheritance

MySQL	Name of an SQL-based relational database management system
n:m	many-to-many (relationship)
NED	Netherlands
NIST	National Institute of Standards and Technology
OM	Optical Microscopy
Inc.	Incorporated
IR	Infrared
ISIS	Name of a tribological expert system developed in the United Kingdom of Great Britain and Northern Ireland
IVW	Leibniz-Insitut für Verbundwerkstoffe GmbH
JSON	JavaScript Object Notation
OOP	Object Oriented Programming
ORM	Object-Relational Mapper or Object-Relational Mapping
PA66	Polyamide 66
PC	Personal Computer
PDF	Portable Document Format
PEEK	Polyether ether ketone
PHP	Perl Hypertext Preprocessor
PM	Project Management
PostgreSQL	Name of an SQL-based relational database management system
PPS	Polyphenylene sulfide
PRECEPT	Name of a tribological expert system developed in the Netherlands
PTFE	Polytetrafluorethylene
RAID	Redundant Array of Independent Disks
(Ruby on) Rails	Web development framework for Ruby
RDBMS	Relational Database Management System
Ruby	Name of an object-oriented programming language
rH	Relative Humidity
rpm	Revolutions Per Minute
SEM	Scanning Electron Microscopy
SOVTRIBO	Name of a tribological LIMS developed in the Soviet Union

SQL	Structured Query Language
STEM	Scanning Transmission Electron Microscopy
STI	Single-Table Inheritance
TB	Terabyte
TLA	Transfer Film Luminance Analysis
TR4	Mainboard socket for CPUs
TRIBEX	Name of a tribological expert system and database developed in the Federal Republic of Germany
TRIBOCOLLECT	Name of a tribological LIMS developed in the Federal Republic of Germany
TRIBODATA	Name of a tribological database developed in the Federal Republic of Germany
TRIBOLOG	Name of a tribological database developed in France
TRIBSEL	Name of a tribological expert system developed in the United Kingdom of Great Britain and Northern Ireland
TS	Test Specimen
TSC	Test Specimen Class
TSL	Test Specimen Lot
UAP	Uniform Access Principle
US(A)	United States (of America)
UK	United Kingdom of Great Britain and Northern Ireland
UML	Unified Modelling Language
UMT-3	Universal Mechanical Tester 3, brand name of a tribometer manufactured by Bruker Corporation
URL	Uniform Resource Locator
USSR	Soviet Union
UTC	Universal Time Coordinated
UTF-8	8-Bit Universal Coded Character Set Transformation Format
XML	Extensible Markup Language
YAML	Yaml Ain't Markup Language

Latin symbols

$^{\circ}\text{C}$	Degree(s) Celcius
$\text{\d{n}}$	n-digit integer
\s+	Whitespace, one or more
A	In-roi horizontal coordinate
a	Horizontal image position
a_i	Horizontal beginning of the region of interest, in absolute image coordinates
a_f	Horizontal end of the region of interest, in absolute image coordinates
$\mathcal{A}\text{-}\mathcal{H}$	Beam path designations for TLA
B	In-roi vertical coordinate
b	Vertical image position
b_i	Vertical beginning of the region of interest, in absolute image coordinates
b_f	Vertical end of the region of interest, in absolute image coordinates
f	Aperture number
h	Hour(s)
K	Calibration constant of a camera's light sensor
km	Kilometer
L	Luminance, absolute, w/ polishing correction
l	Luminance, absolute, w/o polishing correction
mm	Millimeter
N	Grayscale pixel value
n	Integer number
P	Test parameter for polishing detection
R	Polishing ratio

R_a	Arithmetical mean deviation of the assessed profile, a roughness parameter
R_z	Average of the maximum peak to valley heights of the various sampling lengths of a surface profile
R_p	Average of the maximum valley peaks above the mean line of the various sampling lengths of a surface profile
R_v	Average of the maximum valley depths below the mean line of the various sampling lengths of a surface profile
S	Sensitivity of a camera's light sensor
t	Time
t_{exp}	Exposure time
t_{rot}	Rotation period
w	Wear track width
x	Wear track lateral position

Greek symbols

α	Threshold for polishing detection
ΔA	Width of the region of interest
ΔB	Height of the region of interest
Θ	Heaviside function

Mathematical symbols, Indices, Superscripts, Miscellaneous

#	Comment in Ruby code or notation for an instance method
Σ	Sum
*	Reference area
TM	Trademark
%	Percent
roi	Region of interest
rel	Relative
ref	Reference area
avg	Average
pol	Polishing detection region
ref-pol	Reference region for polishing detection

1 Laboratory Information Management in Tribology

Laboratory information systems (LIS) are mostly defined by their purpose and functionality: “They were initially developed to collect, record, present, organize, and archive laboratory results, often with a focus on generating information for proper financial management of the laboratory.” [1]. While LIS focus on test specimen and testing data, Laboratory Information Management Systems (LIMS) extend this to data analysis, workflow and features to meet regulatory requirements. In 2012, Prasad et al. provided a definition that reflects the extended scope of a LIMS over a mere LIS: “In a very broad sense, the term IS is frequently used to refer the interaction between people, processes, data and technology. A management system is the framework of processes and procedures used to ensure that an organization can fulfill all tasks required to achieve its objectives” [2]. A possible use-agnostic definition of a LIMS might be that it is a data system that manages information that are created and processed in a laboratory environment and that additionally supports a given set of operations in order to comply with internal and external regulations. Despite the difference in scope, the terms LIS and LIMS are frequently used as synonyms.

1.1 State of the Art

The use of information technology in tribology can be divided into four different types of application: test result and material property databases, bibliographical databases, LIMS and expert systems.

Databases for test results and material properties as well as bibliographic databases are often designed as “flat file databases”, i.e. as single tables which have a large number of columns which contain the individual pieces of information on test parameters and results [3]. As the results of tribological experiments depend on a wide range of parameters, “like load, velocity, temperature, type of motion, contact geometry, surrounding medium, the compilation of a numerical database for tribological results must include information on the test parameters and the material treatment of the test couples” [4]. The user benefit of such systems originates from the very quick retrieval of large amounts of data from one single source and from the level of detail that can be crafted into electronic queries, e.g. with respect

to type of material, temperature range etc. Of course, both of these benefits significantly depend upon the amount of data stored and on the efforts that were put into the original system design and into data input. While easy to implement and maintain, flat file databases are highly prone to inconsistencies that arise from spelling errors when entering redundant data. Keeping the same information that is stored in different locations (rows) of the table synchronous over subsequent batch updates is another issue for such databases. Furthermore, as the number of rows grows, the potentially high degree of information redundancy takes up much memory and slows down database operations as more information needs to get parsed. The main advantages of flat file databases are their ease of design and implementation. However, due to their issues with high data redundancy, reduced performance and low data integrity they are usually only used for small projects with limited scope (e.g. spreadsheet calculations) where short development time greatly outweighs these disadvantages. For large scale, long terms projects with compliance requirements, flat file databases are no viable option.

Tribological expert systems “consist basically of a combination of literature databases, material databases, construction rules and numerical results databases” [4]. Thus, they usually are complex software packages that comprise tools for analyzing tribological systems with respect to loadings, temperatures, shear stresses, contact mechanics and other parameters that occur in the system in question. Based on such analyses, they offer tools for the design and basic numerical simulation of typical tribological components of which journal bearings are the most prominent example.

LIMS were originally developed for use in medicine, analytical laboratories in health care, biology and chemistry. This origin is reflected by the results of a Scopus search on KEY(“Laboratory Information System” OR LIMS). On 28th Nov 2021, this query yielded 2,810 hits. Amongst others, they were distributed across subject area as follows: Medicine 2,053, Biochemistry, Genetics and Molecular Biology 496, Health Professions 263, Computer Science 245, Engineering 213. Even when restricting the search results to “Engineering”, the spectrum of keywords revealed that the main focus still was on medical disciplines: Laboratory Information System 88, Clinical Laboratory Information Systems 75, Human 57, Hospital Information System 52, Humans 50, Laboratories 49. Changing the above query to include

“Tribology” in either Title, Abstract or Keyword field yielded no hits at all. The same was true for adding “Wear” to these fields instead of “Tribology”.

The first notable effort to use modern computer and information technology in the field of tribology dates back to 1985 when the National Institute of Standards and Technology (NIST) of the United States of America started the development of “A Computerized Tribology Information System” (ACTIS) [5], [6]. Its aim was to concentrate tribological data in one location and therefore simplify the transfer of latest tribological research into application. It not only contained tribological data like wear rates and coefficients of friction but also other material data like heat capacity, density or electrical resistivity. Technically, ACTIS consisted of a flat file database with 43 columns and 368 rows (entries). The dataset that was provided with ACTIS was selected from scientific literature by the software maintainers in cooperation with tribologists. Selection of material candidates for a given application was then achieved by entering ranges or limits for parameters like wear rate into an application software that was run on a PC. Based on user entry, the software then filtered the database entries and retrieved the entries that matched the user-supplied criteria. Beyond being a sole material property database, ACTIS also contained expert system modules that should support the design and numerical modeling of basic tribological components like plain and journal bearings as well as gears [7]. ACTIS was marketed from 1994 by Actis Inc. of Wilmington, Delaware, USA [8]. However, as of 2021, ACTIS Inc. has ceased operation and ACTIS is of no further relevance.

Roughly at the same time than ACTIS was developed in the US, A.M. Zhakarov of the Russian Academy of Sciences started to develop SOVTRIBO [7]. It consisted of a bibliographical index, tribological material data and expert systems for the design of tribological components required for the construction of agricultural machinery as well as for calculating wear between rail tracks and train wheels [5]. Furthermore, SOVTRIBO also contained databases and modules that were developed and maintained by other research institutions of the former USSR, specifically on material data for polymers and composite materials and material data from low temperature and vacuum tribology testing [7]. Overall, there is only very little information about SOVTRIBO available in Western literature, especially there are no referrals to it anymore in recent literature. Hence, it must be assumed that SOVTRIBO, like its US counterpart ACTIS, is of no more relevance today.

Table 1: Historical and existing electronic data systems for material and tribological data, reproduced from [9], expanded

Acronym	Title	Country	System type/content
ACTIS	A numerical tribology information system	USA	Expert system, data base
PRECEPT	Tribological principles in an expert system	NED	Expert system
ARMES	Abrasion resistant materials expert system	AUS	Expert system
TRIBEX	Wear expert system for unlubricated tribosystems	GER	Expert system, data base
TRIBOLOG	Numerical data bank	FRA	Data base
TRIBODATA	Tribological behaviour of polymers	GER	Data base
TRIBSEL	Coating selection expert system	UK	Expert system
ISIS	Surface coating selection system	UK	Expert system
TRIBOCOLLECT	Numerical, tribological data base	GER	Data base
SOVTRIBO	SOVTRIBO	USSR	Bibliographical index, expert system, results data base
- none -	Data-Base System of Tribological Coating Materials	Jordan	Data base
CTDB	China Tribology Database	CHN	Publications, Products, Tribologists and Research Units, Material data, Greases, Hydrodynamic bearing

A much more recent example of a tribological test result and material property database is TRIBOCOLLECT which has been designed and developed by a team of researchers of the Bundesanstalt für Materialforschung (BAM) in Germany [9]. It is designed as a flat file database that “contains about 150 attributes per set of data (numeric and alphanumeric)” [4].

In 1993, work on the China Tribology Database (CTDB) began. It consists of several subsystems, including a general information subsystem that consists of “four sub-banks: Publications, Products, Tribologists and Research Units”. Furthermore, there is a technical subsystem that contains “data of friction materials, abrasive wear with fixed abrasive, abrasive wear with loosen abrasive, static friction coefficients, friction and wear under boundary lubrication in reciprocating motion, seizure (PV) limits, fatigue wear of slide bearing materials under boundary lubrication. It is also the data of erosion of metals, viscosity–pressure and viscosity–temperature

relations of lubricating oils produced in China, and anti-friction or anti-wear performance of materials under reinforced friction and wear test.” Furthermore, data on greases and on “stiffness and damping coefficients of hydrodynamic bearings” are included. CTDB is intended to “connect with the China Modern Design Net (CMDN) to support a plan of remote collaborative design in China.” [10] As of December 2021, no scientific literature on CTDB other than [10] can be found in bibliographical databases.

In 2008, Sedlaček et al. reported that they had developed a database for tribological properties of Diamond Like Coatings (DLC) in order to be able to collect and compare data on DLC coatings, to improve the process of selecting a coating for a given set of requirements and to gain new insights by analyzing the collected data. [11] For this, they designated 49 attribute fields, including fields on material description, coating deposition parameters, test specimen dimensions, test parameters and test results. The authors did not only put their own test results into their database but also retrieved data on about 800 coatings from scientific literature. Although they only took their data from selected peer-reviewed, high-impact journals (e.g. Surface and Coatings Technology, Wear, Material Science and Engineering, Tribology International and Thin Solid Films), where “testing procedures and conditions as well as the results and conclusions are described”, they found that “in spite of this, it is not always easy to compare the results” or to “make constructive comparisons”. Specifically, they find that the lack of a common reporting standard within the scientific community resulted in a large, albeit inhomogeneous, data set: “The main problem in gathering data from the bibliography lies in the fact that the results are largely given in very general, sometimes only informational terms. Furthermore, the test conditions are often not defined exactly. Thus the coefficient of friction values are given in large spans (from 0.05 to 0.2), with the degree of wear too often described as great, small or hardly measurable. This kind of data is useless if we want to conduct any kind of comparison.” Therefore, their attempt to correlate results of their own pin-on-disc tests on DLC coatings with the data from their database yielded only very limited correlation for the coefficient of friction, while the prediction of the impact of contact conditions on friction and wear was virtually impossible. As of December 2021, no follow-up publications on the 2008-paper of Sedlaček et al. can be found, and although, according to Scopus, it

has been cited 55 times, citing papers mostly refer to the numerical results presented in the paper in order to compare them with their own results. In 2012, Abu-Ein et al. reported on their database for tribological “coatings, their properties, composition, and applications” which is very similar to Sedlaček’s approach [12].

In 2019, work began on the most recent electronic system for tribological data called i-TRIBOMAT. Its goal is to “develop the world’s largest user-driven open innovation test bed that enables versatile tribological characterisation of materials and the up-scaling of tribological material behaviour to industrial dimensions” [13]. It aims to do so by establishing a “shared tribological infrastructure of more than 100 tribometers and characterisation equipment”, including “newly developed protocols ... and online data acquisition”. It is intended to provide “services (for) data storage, data sharing and data analytics” as well as a “tribological material database”. Additionally, i-TRIBOMAT should include expert systems for component design which are supposed to bridge the “gap between lab-scale characterisation of materials, and the tribological requirements of a real-life application”. This is supposedly achieved by providing “material models and dynamic solvers to predict the system-dependent tribological behavior of materials, like the life-time or energy efficiency of various machine components”. Ultimately, i-TRIBOMAT aims at finally enabling the transferal of “laboratory results to field applications (lab-2-field)” [14]. In the context of the work on i-TRIBOMAT, the “challenges for the design of a universal tribological database for materials” have been formulated [15] of which data harmonization, data interrelation, data re-traceability, data searchability, metadata storage and results reporting are the most critical issues to be solved by i-TRIBOMAT. The suggested solution for these issues is to utilize the commercial and proprietary Granta MI software package of ANSYS Inc. While including a renowned company in i-TRIBOMAT probably increases the chance to achieve the project’s ambitious goals, it must be seen how much of the underlying technology will be disclosed to the academic community. As of December 2021, only a list of “Challenges for the design of a universal tribological database for materials” has been published [15] but nothing about the solutions that have been identified at least on the design or the actual implementation of the data model for laboratory or field experiments [16].

In 2021, Zhang et al. described a general concept of “tribo-informatics” and an architecture for it [17]. They define “tribo-informatics” as a collection of interrelated

tasks establishing tribology standards, building tribology databases, and using information technology to collect, classify, store, retrieve, analyze, and disseminate tribology information with the aim of increasing the research efficiency of tribology. They too note that the establishment of a comprehensive tribological database is much more difficult than for other disciplines (like chemistry, biology or mechanical properties) since tribological systems contain many different components (like coatings or lubrication oils) that all need to be described properly in order to ensure the usability of collected data. The architecture that the authors describe ranges from storing experimental raw data through data analysis and interpretation to the inclusion of stored data in theoretical models, simulation models and artificial intelligence prediction models. However, the authors only provide a general description of a “tribo-informatics” architecture. Although they provide a use case in which an implementation of such a system has proven useful to develop and optimize computational models that predict the impact of test parameters on friction and wear of a bushing, no information is provided about which actual components should form such a tribo-informatics system or how it should actually work.

Already in 2020, Kügler et al. have addressed the issue of the conceptual complexity of tribological experiments and described an ontology for ball-on-plate tests which gives a more formal definition on an important range of entities that are part of tribological experiments [18]. While this can be considered a “first step of a formal and explicit specification to enable a shared understanding in the domain of tribology” and opens up the possibility of for the “application of machine learning techniques for automated processing and analyzing data”, it still does not answer the question on how use such an ontology to provision support for everyday operations in a tribology laboratory, especially to non-experts in the field of ontologies. Overall, tribology laboratories conduct most of their procedures manually. Especially defining individual tests and whole test plans for projects, labelling test specimen, handling and evaluation of measurement data files, result aggregation and statistical analysis as well as plotting of primary, computed and aggregated data. Some tribometer manufacturers provide proprietary solutions for use with their equipment that provide computerized assistance for selected tasks, most notably Mooha™ from Ducom Instruments [19] and TriboScript™ from Bruker Corporation [20]. However, being proprietary software, they are typically not free to use, not interoperable and limited to functionality of the manufacturers’ tribometers.

1.2 Shortcomings of the state of the art, objectives of this work and solution procedure

Until today, no universally accepted LIMS exists for tribology laboratories beyond some narrow-scoped proprietary solutions. Existing solutions and current efforts are mainly results databases which serve as downstream data storage and therefore play practically no role in actual laboratory operations. In the area of specimen management, there is a lack of information systems that support critical laboratory processes such as the assignment of identifiers for materials and specimens, the storage of specimen characteristics at different aggregate levels (individually or as a batch), or the systematic archiving of specimens. In the area of test creation and administration, there is currently no uniform information system for the large variety of test types, test specimen arrangements, movement types and test procedures. In the area of test execution, the main challenges are the lack of automated processes for the parameterization of tribometers and the administration of primary test data. The degree of computerization in the evaluation and plotting of measurement data and in their aggregation is correspondingly low.

The objective of this thesis is to bring the domain knowledge of operating a tribological laboratory, data modelling and programming together in order to create a LIMS that provides solution for these shortcomings. Specifically, it will demonstrate how to build data models for all key entities of a tribology laboratory and how to use using object-oriented programming for implementing them into a LIMS that meets the main data storage and process support needs of tribologists. An additional aim is to demonstrate how using such a system can increase data integrity and quality over what is achievable by manual processes. To make sure that that its results are universally applicable by the tribological research community, it only uses open-source software that is licensed “for free” for academic use.

For this, this thesis first lays out the fundamental concepts and technologies that are needed for translating domain entities into data models and for writing a server-run, browser-interfaced web application (Section 2, Fundamentals). Then, based on a workflow analysis, a set of requirements for such a LIMS is formulated (Section 3, Requirements for a LIMS for tribology laboratories). Section 4 then presents *Atlas*, a sample implementation of a LIMS for tribology laboratories. Finally, in Section 5, Transfer Film Luminance Analysis (TLA) is presented. While it solves the most pressing issues of other techniques for investigating transfer films, it demands

the evaluation of a large number of images, the synchronization and joining of different data sources and the persistence of metadata and a large amount of resulting data that it cannot be executed manually. However, by integrating it into the newly developed laboratory information system, it succeeds safely, quickly and reliably traceable in the long term.

2 Fundamentals

While formally also being state of the art, this section summarizes concepts and techniques that are used as tools to achieve the desired scientific progress in the field of LIMS for tribology laboratories.

2.1 Entity-Relationship Data Modeling

The entity-relationship (data) model (“ER model”, also “ER data model”) has been formally described by Chen in 1976 [21]. ER modeling is the structured process of creating an ER data model for a given scope (also called “domain” or “business domain”). A data model mainly consists of entity types, entity attributes and their relationships as well as a set of rules, often called “domain logic” or “business logic” that apply to entity classes and their interaction, as well as of the relationships that exist between different entity classes.

According to Chen, “An entity is a ‘thing’ which can be distinctly identified. A specific person, company, or event is an example of an entity” [21]. Entity classes are abstract definitions used to classify individual entities. Examples of entity classes that can be used in a tribology laboratory are “block test specimen” (a physical object), tribological experiment (a concept) or test specimen machining (a process). A specific tribometer, i.e. an “UMT-3” (by Bruker Corporation), is an example for an entity that belongs to the entity class “tribometers”.

A relationship, according to Chen, “... is an association among entities. For instance, ‘father-son’ is a relationship between two ‘person’ entities.” The description of relationships includes, first, names of the associated entity classes. However, as in this example, actually only one single entity class – the person class – is involved in the “father-son” association. Therefore, both entities should be given role names that clarify the role of the individual class. In this case, the “person” class participates in the “father-son” relationship in two roles: “is father of” and “is son of”. In 1983, Chen published a guide for extracting entities, their classes and their roles from English sentences [22]. Finally in 1997, Chen suggested the use of fixed word types for various aspects of associations: common nouns for entity classes (“planet”), proper nouns for specific entities (“jupiter”, “solar system”), transitive verbs for roles (“is part of”), intransitive verbs for entity attributes (“large planet”) etc. [23]. This system is still in use today and is the de-facto standard in

naming relationships and their components. Furthermore, relationships are characterized by two meta-attributes: optionality and cardinality. Cardinality describes the maximum number of entities that can be part of the relationship (for both roles). If a relation is optional (on any given role), the number of entities that can take fulfill the respective role can be zero. Typically, cardinalities larger than one are not specified by a finite number but by “many”. Therefore, the following standard cardinalities exist: one-to-one (1:1), one-to-many (1:n), many-to-one (n:1) and many-to-many (n:m), with n and m being potentially infinite positive integers, as well as their optional analogues, where n and m are potentially zero. By defining (and enforcing) cardinality and optionality of relationships, business and data integrity rules can be modeled. As an example, by defining that the relationship between the “block on ring-test” class and the “ring tests specimen” class is a mandatory n:1 relationship, instances of this test type can be formally classified as valid (if each test has exactly one test specimen related to it) or as invalid (if there are more than one blocks on any test or if any test does not have an associated block).

Data models exist in three hierarchical levels: conceptual, logical and physical data model. The conceptual data model is the most abstract view on the domain. Its scope typically spans the whole organization, e.g. a research institute, or academic field. Based on this, the logical data model is designed. The logical data will typically have a narrower scope, i.e. friction and wear testing in an institution’s tribology laboratory, and it will contain specific definitions of this scope’s entities (materials, tribometers, test specimen), their attributes (test specimen width), attribute data types (numeric, string, etc.) and the (semantic) relationships between entities. Therefore, several logical data models can exist in an organization or in an academic field in parallel. Finally, the physical data model defines the actual implementation of data storage, typically in a relational database management system (RDBMS, “relational database”). It specifically describes tables to represent entity classes which includes the definition of table columns that store the attributes of an entity class. While the logical data model is abstract enough to be independent of the actual implementation of the physical data storage, the physical data model specifically depends on it and needs to be adjusted in case of changes in the utilized software and/or hardware.

An Entity-Relationship-Diagram (“ERD”) is a graphical representation of an ER model. ERDs use different graphical representations for each component of a data

model, e.g. for entities and relationships. Today, many different ERD “styles” exist, among them the Chen-Notation [21], the Bachman-Notation [24], the Barker-Notation [25], the Martin-Notation [26], [27] and the Unified Modelling Language (UML)-Notation [28], [29]. Mostly, entities are represented by boxes that contain the entities name as a heading and below it a list of attributes. Relationships are represented by arrows which are styled or annotated to represent optionality and cardinality. Barker and Bachmann are basically the same: while Barker uses the “crow’s foot” to denote an n-ary cardinality of a role, Bachmann uses an arrow. Unary roles are simple lines in both notations. Optional roles are denoted with an open circle by Bachmann and by a dashed line by Barker. Martin represents cardinality with a “crow’s foot” for n-ary roles and with a single cross-line for unary roles. On both inner sides of the connections, Martin denotes optionality: an empty circle represents optional roles (like in the Bachmann-notation) but mandatory roles are denoted by a single cross-line. Therefore, mandatory unary roles are represented by two cross-lines in the Martin-notation.

2.2 Object oriented programming

2.2.1 Functional programming

In functional programming (FP) data and functions exist separately within a computer program. Data is typically defined as a set of variables that contain information. Functions are portions of callable code and are intended to operate on data. It is up to the programmer to achieve a given goal by feeding data to the respective functions and to decide what to do with the function output.

In Ruby, the programming language that has been chosen for the implementation of a sample LIMS (see Section 4) functions are called procedures. While some programming languages differentiate between functions, which have one or more return values, and procedures which do not have a return value, Ruby does not make this difference because in Ruby, all callable objects always have a return value (either an explicit one, defined by the `return` keyword, an implicit one, defined by the last evaluated expression or the default return value `nil`). When using an explicit return value, Ruby procedures return not only from themselves but also from the context in which they were called which is often counterintuitive to how a function should behave. However, Ruby procedures have a special subtype called

lambdas which only return from themselves when using an explicit return value. They are therefore the closest match to functions of other programming languages and are therefore often called “Ruby functions”.

For two liquids, water and ethanol, both stored at -14.7 °C (at normal atmospheric pressure) a program should decide whether each is frozen. Code listing 1 shows corresponding Ruby code in functional programming style. First, the lambda `frozen` is defined, then the data that represents the substances’ freezing points and the current temperature is defined and finally, the lambda is called for each substance.

Code listing 1: Deciding whether water and ethanol are frozen at -14.7 °C using functional programming in Ruby (`=>` denotes return values).

```
# function definition
frozen = lambda{ |temp:, freezes_at:| temp <= freezes_at }
# data definition
water_freezes_at = 0.0           ethanol_freezes_at = -114.1
water_boils_at = 100.0          ethanol_boils_at = 78.4
current = -14.7
# function calls
frozen.call(temp: current, freezes_at: water_freezes_at) => true
frozen.call(temp: current, freezes_at: ethanol_freezes_at) => false
```

2.2.2 Object definition

Object oriented programming (OOP) eliminates this separation of function definition, data definition and function calls by joining semantically related data and functionality into a data structure called “object”. Code listing 2 shows the implementation of the same problem in OOP-style Ruby code. In Code listing 1, the two liquids do not appear as monolithic objects in the code. Instead, they are described indirectly by a collection of variables that share the substance name as a prefix, thereby creating a pseudo-namespace (which could of course be refactored into in a composite data structure like a hash, for example, which would also create a semantically joint storage for information on a given object). In Code listing 2 however, each liquid is explicitly represented: one by the variable `water` and the other by the variable `ethanol`, which are instances of either the `Water` or the `Ethanol` class, which are subclasses of the `Substance` class. When instantiated, they are given their respective freezing points which they store internally and which thereby become “attributes”.

Code listing 2: Deciding whether a liquid is frozen or not, OOP style.

```
class Substance
  # instance method definition
  def initialize(freezes_at:)
    @freezes_at = freezes_at
  end
  def frozen(current)
    current < @freezes_at
  end
end
water = Substance.new(freezes_at: 0.0)
ethanol = Substance.new(freezes_at: -114.7)
water.frozen(-14.7) => true
ethanol.frozen(-14.7) => false
```

In this program, as it does not define an attribute setter for `@freezes_at`, the freezing points of instances of `Substance` or its subclasses cannot be altered by direct attribute assignment, see Code listing 3, or by mass assignment. This behavior is called “encapsulation” and is a major feature of OOP that enables granular control on how data stored inside objects can be read or modified.

Code listing 3: Attributes and encapsulation – freezing temperatures defined at object creation cannot be changed later

```
water.freezes_at = -50.0
=> NoMethodError (undefined_method `freezes_at' for 1:Substance)
```

While in FP, the freezing of liquids was defined in a separate function, OOP moved this function into a `Substance` class and called it a “method”. Methods are therefore functions that are defined within a class. While in FP, the `frozen` function is called in the global scope of the program, methods are called on their respective objects. As each of the two objects that represent water and ethanol stores the freezing point of its respective substance internally (in an instance variable), both can execute the same instance method, however each using its own internally stored freezing point. The difference is that, once the objects are instantiated, the caller does not need to know the freezing point of the object but only needs to provide the temperature for which the information on the aggregate state is required. Overall, FP and OOP distinguish themselves significantly by their organization of data and functionality. While FP typically defines a set of functions to operate on data, OOP joins data and functions to form objects whose internal state can be isolated from the exterior.

2.2.3 Inheritance, Generalization and Specialization

Besides encapsulation, another feature of OOP that clearly distinguishes it from FP is inheritance [30]. Inheritance is the construction of classes based on one or more existing classes instead of building each class from scratch. A class that is created this way is called “subclass” and the class from which it is derived is called “superclass”. The central principle of inheritance is that subclasses inherit the attributes and methods of their superclass(es). While other programming languages allow subclasses to be derived from more than one superclass (“multi-class inheritance”, e.g. Python), in Ruby classes can only inherit from one class, but each class can have an unlimited number of subclasses. While allowing single-class inheritance only greatly restricts the flexibility of composing tailored classes from a collection of base classes, Ruby allows classes to mix-in an infinite number of modules (= non-instantiable classes) in addition to inheriting from one class.

Code listing 4: Class inheritance in Ruby

```
# define superclass
class Substance
  def frozen_at(temp)
    temp <= self.class.instance_variable_get(:@freezing_point)
  end
end
# define subclasses
class Water < Substance
  @freezing_point = 0.0
end
class Ethanol < Substance
  @freezing_point = -114.7
end
# object instantiation
water = Water.new
ethanol = Ethanol.new
# method calls
water.frozen_at(-14.7) => true
ethanol.frozen_at(-14.7) => false
```

Code listing 4 revisits the frozen liquids problem but this time three classes are defined. Now, `Substance` only defines the freezing behavior of a substance in general. The specific behavior of water to freeze at 0 °C is now defined in `Water`. In terms of data modelling, `Water` and `Substance` are in a subclass-superclass relationship. The same applies for `Ethanol`.

While the term inheritance implies a downstream flow of object implementation, it can also be seen as an upstream process: OOP allows for groups of similar classes

to be generalized by extracting duplicate behavior and attributes into one or more joint superclass(es). Therefore, inheritance is also called generalization when seen as an upstream process and specialization when it is seen as a downstream process. In each case, the inheritance relationship between superclass and its subclasses is characterized by being either partial and complete and by being either overlapping or disjoint [31]. It is called “disjoint” if an instance can belong to only one subclass and “overlapping” if it can belong to multiple subclasses. The relationship is called “partial” if an instance of the superclass can exist without also belonging to any of the superclass’ subclasses and “complete” (or “total”) if it cannot. For Ruby, inheritance is partial and disjoint. For Ruby on Rails, it can be made complete and disjoint by declaring the superclass to be abstract, i.e. non-instantiable.

2.2.4 Extension and method overriding

In addition to inheriting all attributes from their superclasses, OOP subclasses in general - and particularly Ruby’s subclasses - are extensible which means that subclasses can be given methods that their superclasses do not possess. For example, this is the case when not all methods are extracted into a superclass during generalization or when - when inheriting - subclasses are given methods that are not present in the superclass.

A special form of extension during inheritance is method overriding which in OOP describes the ability of a subclass to provide a specific implementation of an inherited method. Code listing 5 shows implement method overriding in Ruby using the example of the sublimation of carbon dioxide.

Code listing 5: Method overriding

```
class CarbonDioxide < Substance
  def frozen_at(temp)
    false
  end
end

co2 = CarbonDioxide.new
co2.frozen_at(-14.7)
=> false
```

Overriding the `frozen_at` method of the `Substance` superclass results in a situation where different subclasses, in this case `Water`, `Ethanol` and `CarbonDioxide`, exhibit different implementations of the same method name, in this case `frozen_at`.

2.2.5 Association, Aggregation, Property Propagation and Duck Typing

Association is a relationship type that connects entities that are considered to be otherwise independent of each other. Chen's example of the relationship between father and son is such an association. It connects two people that are otherwise independent entities and defines the semantics of this connection.

Aggregation is a special case of association in which one entity is declared to be a part of another entity. An example for an aggregation is a collection of people that are students of the same class. Frequently, aggregation objects consist of collections of objects that are similar in type, attributes and methods. For an aggregation object, emphasis is not on its own attributes or methods but on those of its members. Aggregation objects (which sometimes are also called association objects) typically implement methods that do not access internally stored attributes of the aggregation object itself, but whose return value is derived from attributes and methods of their members. This technique is called property propagation. The algorithm which determines how exactly the response is derived from the members is called propagation function [32]. Code listing 7 shows how a surface object is composed of polygon objects of different types and how the total area of a surface is derived from the individual members by summing up their respective areas. Therefore, the area property of a surface object is a propagated property. This OOP design pattern, where all potential members of a collection implement a certain method (or set of methods) so that they can be iterated over without explicit type checking is called "duck typing" (refer to [33] for a more formal definition).

Code listing 6: `Surface` as an aggregation class whose instances propagate their `area` property.

```
class Surface          class Rectangle          class Square
  def initialize(members:)  def initialize(a, b)  def initialize(a)
    @members = members    @a = a; @b = b      @a = a
  end                    end
  def area              def area              def area
    @members.map(&:area).sum  @a * @b              @a * @a
  end                    end
end                    end                    end

r = Rectangle.new(a: 7, b: 2)
s1 = Square.new(a: 3)
s2 = Square.new(a: 4)
Surface.new(members: [r, s1, s2]).area => 39
```


2.2.6 Composition

Another important concept of object-relational data modelling and of OOP is composition which has been initially described by Smith in 1977 [23] and which composes objects from two or more other objects, called components. The difference to aggregation objects is that the existence of a composition's components depends on the existence of the composed object. When a university gets dissolved, its individual departments cannot exist independently and get dissolved as well. In contrast to aggregation objects, emphasis is on attributes and methods of the composite itself.

2.2.7 Uniform Access Principle

The Uniform Access Principle (UAP) says that “all services offered by a module should be available through a uniform notation, which does not betray whether they are implemented through storage or through computation” [34]. For a programming language to fulfill this requirement, all requests to an object, particularly method calls and attribute getters must be syntactically equivalent. This enables the internal implementation of a response to any given call to be effectively made opaque to the world outside of an object. A prominent example is that attributes can be either internally stored data that can be simply retrieved and returned, or it can be a virtual attribute, whose value is not actually persisted, but which is the result of an operation that is performed when the attribute is called. The same concept applies to setting values of attributes. Therefore, programming languages that support UAP, like Ruby, greatly simplify the construction of aggregation objects with type-mixed members who partially exhibit stored and virtual attributes. Furthermore, they enable code that is much more extensible and much more robust with respect to refactoring attributes to methods and vice versa.

2.3 Relational Database Management Systems

A relational database management system is a software that maintains a relational database. A relational database structures data according to the relational data model proposed by E. F. Codd in 1970 [35]. In this model, data sets are stored in tables of rows and columns. The data model is called “relational” because Codd used the mathematical term “relation” to describe a row of data in a table. By default, any given table of a relational database is used to represent one entity type.

Therefore, a data model that includes “materials” and “parts” for example would have a separate table for both types of entities. Columns of a table represent the individual attributes of an entity, e.g. density, name, or color for materials. Furthermore, columns have data types, e.g. “decimal” for density and “string” for color of name. Rows, also called records, then contain individual instances of the respective entity. In order to execute operations on a database, a relational database language is needed. Many relational database management systems use the Structured Query Language (SQL) for this. SQL has been developed by IBM based on the relational data model presented by Codd [35], [36].

2.3.1 Identity

In order to be able to select individual records, records need to have an *identity*, i.e. a set of attributes that unambiguously distinguishes it from other records of the same table. As, mathematically speaking, records are sets, no two records may coincide, i.e. have the same values for all attributes. Therefore, a set of all records is the trivial choice for identity. For every table of a relational database such a set of attributes must be defined and this set is called the *primary key* of that table. This implies that a table with a given set of columns and a given set of rows may have multiple attribute sets that qualify as primary key. While for an empty table any set of attributes is still eligible as primary key, as no records exist that might violate the uniqueness requirement of a primary key, the choice of primary keys can be limited for tables with existing records.

On the other hand, by picking a given primary key for a table, further additions of rows to the table may be limited with respect of which values their attributes can have. Therefore, the choice of a primary key must be made with care and consideration. However, there are scenarios in which duplicate entries, i.e. entries which coincide with respect to all of their attributes that are part of the primary key should be allowed. The typical solution for this is to introduce an attribute that is not actually required to describe the nature of the entity to be recorded but that is solely introduced to enable the discernibility of otherwise identical records, i.e. to ensure their identity. Therefore, this attribute is typically named “id”. It is a typical example of metadata, as it is “data that provides information about other data” [37].

2.3.2 Relationships

Relational databases are able to record relationships between records of different tables. The standard procedure for this is to set the value of a selected column in one table to the value of the primary key of the other table. Such a column is called a foreign key column and the value in any row of that column is called “foreign key”. The table, which contains the foreign key column is called the “referencing table” of that specific relationship and the table whose records are referenced is called the “referenced table”. On the level of individual records, the terms “referencing record” and “referenced record” are used. As foreign keys do need to have unique values, multiple records of one table may reference the same record in another table. In general, a table can have none, one or multiple foreign key columns.

If a record in a given table is thus referenced by multiple records of another table and if a referenced record is updated, the updated information will automatically be retrieved if any of the referencing records navigates the reference to retrieve the referenced record. This way data redundancy can be eliminated, which greatly decreases the required memory while increasing data integrity and performance compared to flat file databases. This greatly simplifies the development of multiple-entity projects. Relational databases are therefore a good choice for implementing Laboratory Information Managements Systems.

In OOP, classes can also have “polymorphic associations” to other classes. In the case of binary associations only one of the two roles is defined statically, i.e. the class that plays that static role is defined explicitly and cannot be changed. In contrast to normal binary associations the second role can be played by a range of different classes. The exact class that plays that variable role can be different or the same for all instances of the fixed class.

While ternary relationships, i.e. relationships with three roles, and even higher-order relationships exist in theory, they have not found widespread use. Instead, they are typically upgraded to entities of their own, for which a collection of binary relationships is then defined [38].

2.3.3 Data Quality and Data Integrity

Data integrity is the maintenance of accuracy and consistency of data. Maintaining accuracy means that data is not altered unintentionally over time or as the result

of repeated read or write operations. Data consistency mainly refers to a system's ability to ensure that data complies to a defined set of rules. These rules originate from the domain logic and must be formulated within the database system with a set of tools provided by it. Examples for tools provided by PostgreSQL are required fields ("NOT NULL"), unique entries ("UNIQUE") and foreign key constraints. When a column of a table is declared as "NOT NULL", no record in that table may contain a "null" value, i.e. no data at all. This is typically used to represent the optionality of an attribute of a data model. When a field (or set of fields) is declared as foreign key, the RDMS ensures that the referenced record actually exists ("foreign key restraint"). Once a relationship that is subject to a foreign key restraint is established, the RDMS will ensure the ongoing integrity of this relationship. In order to do so, it needs to be declared how to handle actions that would break existing relationships. The typical example is that an action tries to delete record B that is referenced by record A. If the delete operation would be allowed to succeed, A would reference a no more-existing record. Possible options to resolve such a situation are a) to reject the delete operation ("RESTRICT"), b) to delete A as well ("CASCADE"), c) delete the reference between A and B (by blanking the value of the foreign key field(s) ("SET NULL"), valid only if the role of A's entity within the relationship is "optional") or d) to set the value of A's foreign key field to a default value (e.g. if the relationship is mandatory on A's side, "SET DEFAULT"). It is up to the stakeholder of the business rules to define the response of the RDMS to any action that will destroy the integrity of any declared relationships.

Data quality is a measure for the usability of data for a given purpose. Therefore, important aspects of data quality are completeness and correctness. In contrast to data integrity, due to their complexity and variability, these two properties of data cannot be easily ensured by database systems. However, there is a small overlap between data integrity and data quality: business rules be designed to impose constraints on quality-related information: mandatory fields, data types or duplicate prevention can be a great help in ensuring that the users of system obtain information that is considered critical to data quality and to convert to the most useful data type before entering it into the system. Also, on the application level, transforming text type fields into select fields which get populated by the instances of

another model directly relates to the use of a foreign key constraint (with the classical example being a select field instead of a text field for the “country” attribute on an address).

2.4 Object relational mapping

In OOP programming languages, entities and their classes are represented by in-memory objects. RDMS, however, store information as tuples in tables. Therefore, OOP objects and database records are incompatible data types. This disparity of concepts (objects with attributes versus tables with fields and rows) is called “object-relational impedance mismatch” and is by itself the subject of academic research [39]–[41]. When in-memory are to be persisted, a statement must be formulated that can be interpreted by a RDMS and that, upon successful execution, results in the creation of a new database record or in the update of an existing record. On the other hand, when an in-memory object is to be loaded from storage, a statement must be formulated that can be interpreted by a RDMS and that makes it return the corresponding record. Then, the returned information must be transformed into an in-memory object. Web application frameworks typically include subsystems, called “object-relational mappers” (ORM) that handle this conversion processes automatically. Rails, for example, ships with an implementation of the ActiveRecord ORM design pattern invented by Fowler [42].

2.5 Representation of specialization in RDMS

There are two main strategies to reflect object inheritance on the level of a RDMS. First, the instances of sibling-classes, i.e. subclasses of the same superclass, can be persisted in one single table. This approach is called “single table inheritance” (STI) and is typically used when subclasses differ mainly by behavior, i.e. by method implementation, but otherwise use the same attributes. This results in relatively few empty database fields, as only a small number of columns is used by only a few subclasses. In the opposite case, i.e. when siblings use highly disjoint sets of attributes, multi-table inheritance (MTI), where sibling classes are stored in a separate tables, is a more feasible approach. For MTI, the declaration of relationships becomes more difficult on the database level, as any record who wants to relate to an instance of an MTI subclass would also need to know the type of the associated record in addition to its primary key. This issue naturally does not exist

for STI where instances of sibling classes are stored in one table. While for MTI the class of a record is unambiguously defined by the table it is stored in, STI requires a type column for specifying the class of each individual record.

2.6 Web applications

The most popular system architecture for web applications is the client-server model, which divides the application's execution domain into client-side and server-side with both being connected via a computer network. Communication happens exclusively between the clients and the server but never between two individual clients. In addition, there is no application logic stored permanently ("installed") on the client side. Instead, whenever a client needs to receive data, it needs to connect to the server and ask for it. When this software architecture pattern was originally designed, clients were considered 'dumb' devices, which should only display data or collect raw data for sending to the server. Processing was done exclusively on the server side. The disadvantage, however, is that every server request induced a network latency when submitting the request, another latency while the server processed the request and then another for transmitting the request response back over the network. With the advancements made in client-side hardware, browser technology and the advent of server-side scripting languages, especially JavaScript, the server-client architecture was slightly modified. It was now possible to not only transmit static information, e.g. HTML pages, to a client for mere display but also to add code that could be executed on the client side. While this also brought along some new issues, particularly security-related ones [43], the number of server requests could be reduced significantly as the client was now able to autonomously react to user actions. The next big step in the development of client-server web applications was the invention of asynchronous request [44]. While in a synchronous request to the server resulted in a complete reload of the entire web page, asynchronous web applications allow for the retrieval of sub-page information snippets from the server and for the partial update of the client's application state without reloading the entire page.

3 Requirements for a LIMS for tribology laboratories

The main task for LIMS is to support laboratory processes. Therefore, this section analyses a typical workflow of a project in a laboratory that mainly does standard compliant (category VI, see [45]) model wear tests, customized model wear tests (category V) and some component tests on components that closely resemble standard test specimens of model wear tests, e.g. thrust washer discs, anti-friction coatings on planar components or journal bearings and bushings (category IV). For laboratories that do higher-category tests, the workflow may look different, specifically about test specimen preparation and the complexity of test specimen mounting and test setup.

3.1 Workflow analysis

Figure 1 represents a typical workflow in a friction and wear testing project. It is simplified because all potential decision-making points have been replaced by fixed routes and alternative routes have been ignored altogether. The process may represent a project as a whole or only a subproject of a larger super project. In this case, the starting point for the whole workflow is the requirement of a customer or of a super project specification. Typical requirements are the selection of a suitable material for a specified application or the production of test data for testing a scientific hypothesis. Furthermore, the reception of material samples is the starting point for a second workflow. The formulation of two separate processes reflects the two different roles that are typically involved in tribological testing projects: the workflow on the left is executed by the project manager, who is typically an academic tribologist (“PM workflow”). This workflow includes tasks that require academic education and experience in the field of tribology: translating customer or project requirements into a test plan, specifying the exact type, number and parameterization of tests, evaluate test results and report generation. The workflow on the right side is made up of a series of fundamental and technical tasks: reception of material samples, test specimen machining, test setup and conduction, test specimen examination and committing test specimen to storage (“laboratory workflow”).

Within the PM workflow, the first step is to design a generic test plan based on the customer or project requirements. Typically, this involves the definition of basic project parameters: type of tests, selection of material combinations, tribological parameters (loading, temperature, sliding speed, lubrication) and whether to do multi-segment or single-segment tests. In addition, whether to use a design of experiment approach or whether to use classical permutation of test parameters, test types and material combination needs to be decided. This sub-process greatly depends on a combination of scientific education and practical experience. Any attempt to digitalize this process either in full or even only partially would have to deal with a large amount of unstructured information. However, potentially within the scope of ontologies and machine learning (see [18]), this is currently out of scope for *Atlas*.

The next two sub-processes of the PM workflow, the type and number of required test specimen and the tests that should be conducted need to be defined. Specifically, this means that instances need to be created which requires the generation of identities, the recording of attributes and metadata as well as the establishment of relationships to many other entities. Examples include the name and potentially the production lot of the material that a test specimen is (to be) made of, its dimensions, the tribometer on

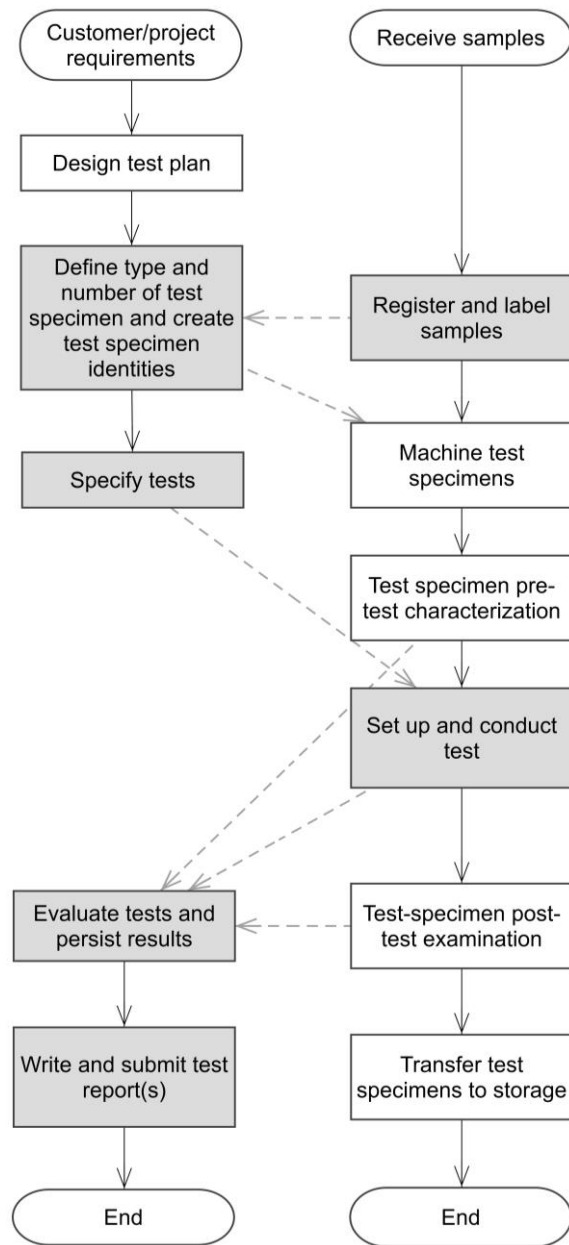


Figure 1: Exemplary project workflow. Boxes represent processes, arrows represent workflow and dashed arrows represent main data/information flow. Test setup and conduction are executed in a loop until all tests have been done. Grey: computerized support.

which a test should be conducted and the project that it should belong to. Furthermore, for the sake of statistical analysis, individual tests are typically parts of a test series that consist of a set of identically parameterized test. These sets need to be created as well, including the establishment of the membership of each test in such a set. These two steps, especially the creation of identifiable test specimen instances rely on information from the laboratory workflow, where received material samples are registered, identified and labelled.

When the tests and the required specimen have been defined, this information needs to be transferred to the laboratory workflow so that test specimen can be machined and examined in their pre-test state. This may include numerical investigations like roughness measurement, profilometric volume measurement, hardness measurement and control measurements of machined dimensions as well as procedures with non-numeric results, e.g. photographic documentation. The results of all these investigations must be passed back to the PM workflow because they might be needed in the evaluation and reporting steps.

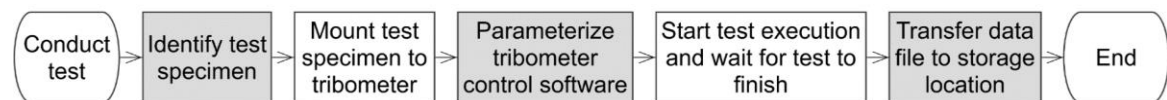


Figure 2: Sub-process for the set-up and execution of a single test. Needs to be iterated for each test. Grey: computerized support.

Figure 2 shows the sub-process for setting up and executing individual tests. It starts with the task of conducting a specific test. Based on the specified test the needed test specimens need to be identified and to be mounted on the corresponding tribometer (which also needs to be identified). The source of information should be the test plan than has been created within the PM workflow when all tests and test set were defined. Based on the same information source, the tribometer control software needs to be parameterized and test execution needs to be started. After the test has ended, the resulting measurement data file needs to be transferred from the local storage on the tribometer to a central storage.

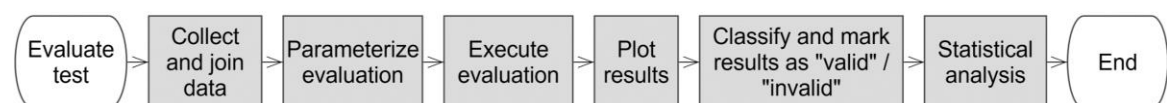


Figure 3: Sub-process for evaluating a single test. Needs to be iterated for each test and test set. Grey: computerized support.

Figure 3 shows an exemplary sub-process for the evaluation of an individual test. First, all relevant data needs to be collected and joined into a single data set. This includes the measurement data file and any auxiliary data files from the tribometer as well as any data from pre-test investigations. If needed, the evaluation process must be parameterized. The classical example for such a parameterization is the selection of evaluation ranges for averaging or fitting data. The evaluation needs then to be executed and its results need to be plotted in order to be able to classify the results as either “valid” or “invalid”. The next step (“statistical analysis”) consists of several of sub-steps: for each test set, based on the numerical results of the corresponding sets’ individual tests, aggregated data needs to be computed. In practice, this means the calculation of arithmetic mean, standard deviation, coefficient of variance, (frequentist) confidence interval and the identification of outliers [46], [47]. These steps must be done for a range of quantities, including linear and specific wear rates, coefficient of friction, various temperatures and ambient conditions. The set of quantities that needs to be analyzed typically depends significantly on customer/project requirements and needs to be specified by the project manager.

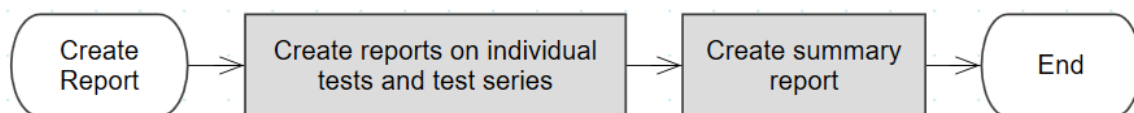


Figure 4: Sub-process for report generation. Grey: computerized support.

Figure 4 shows the sub-process for report generation. It consists of a sub-step for creating test report on test sets which includes the reporting of the computed aggregate data on starts with the plotting of a selected set of primary quantities that have been recorded by the tribometers sensors, e.g, friction force versus sliding distance. Additionally, selected derived and computed quantities might have to be plotted, e.g. coefficient of friction versus sliding distance. Typically, test reports also include aggregated data like the average coefficient of friction in steady state or the linear, time-related wear rate in steady state or the overall test specimen height or weight loss. Metadata also needs to be included in a report with typical data being project name or period of testing.

3.2 Summary of requirements

Based on the above workflow analysis, the main requirements for a LIMS are:

Materials database

- provide a database for materials and their attributes as well as for individual material lots
- manage documents that are related to the reception of material samples

Test specimen management

- provide test specimen ID generation
- persist and manage test specimen attributes (e.g. material name and lot, geometry, dimensions)
- support unique test specimen labeling
- support structured physical storage by providing an indexable and sortable order criterion

Generation of wear tests

- support the process of defining wear tests, including multi-segment tests
- manage relationships, e.g. between projects, test sets and individual tests or between tests and test specimens
- persist test attributes (sliding speed, loading, temperature, ...) on a per segment-basis

Wear test setup and conduction

- provide storage locations for pre- and post-test examinations on test specimens and manage the relationship of examination results to the corresponding test specimens or wear tests
- support the parameterization of tribometers according to the specification of the test to be done, specifically reduce errors due to configuration errors
- support handling, identification and storage of measurement data files, prevent ambiguities that arise from non-uniform data file naming
- automatically extract and persist metadata, e.g. timestamps or tribometer log files

Wear test evaluation

- support the identification and retrieval of the measurement data file(s) that belong a given test

- provide standardized evaluation procedures, enable parameterization and customization
- provide automatic plotting of recorded, derived and aggregated test data
- persist derived and aggregated test data
- persist the manual classification of test data as “valid” or “invalid”, according to [47]
- provide tools for the statistical analysis within a test series, including testing for outliers
- enable and support individual evaluation and data plotting

Compliance, quality management and economic requirements (selection)

Additionally, there are requirements which are workflow-independent:

- enable the traceability of reported data to individual tests and original measurement data files, thereby support internal issue resolving and processing of customer complaints
- prevent manipulation and forgery of measurement data files
- improve data integrity and quality, especially by managing the units of physical quantities
- reduce the error rate for the process of tribometer parameterization

4 Sample implementation: *Atlas*

The LIMS that will be described throughout this work has been given the name “*Atlas*” which is an acronym for its German full name “**A**utomatisches **T**ribo**L**ogisches **A**uswertung**S**ystem” (automatic tribological evaluation system), which represents one of the systems main functionalities.

4.1 Components

Atlas is not a monolithic piece of software but is instead composed of several interacting components. The exact types of its components are a direct consequence of the most basic design decision: *Atlas* is a web application with classical server-client architecture. Therefore, its basic components are: web server, application server, application code and data storage on the server’s operating system. The exact choice of every single component (relational data modelling over tree data modelling, RDBMS over flat table, PostgreSQL over MySQL, Ruby on Rails over Python/Django, etc.) directly affects much of the material that is presented in this work. Still, much of what is presented are concepts. Once established, they can be easily abstracted from the presented implementation and re-implemented in systems with a different set of components.

Throughout this work, implementation of *Atlas* is done in the Ruby programming language, specifically with version 3.0.0 of the CRuby interpreter. Ruby is a dynamically typed, interpreted, high-level open-source programming language that was initially released in 1995 under the “Ruby License” [48] by its inventor Yukihiro Matsumoto and which is still under active development. It was mainly chosen for the Ruby on Rails web application framework that is based on it. For implementing *Atlas*, Ruby on Rails 6.1.4.4 has been used which ships with ActiveRecord as its default ORM [42] and which natively supports OOP concepts that *Atlas* relies on heavily: uniform access principle, duck typing, monkey patching, single table inheritance, polymorphic associations and application-level model invariants. Ruby on Rails is licensed according to the MIT license.

The presented implementation of *Atlas* was done using the open-source database system “PostgreSQL 12”. PostgreSQL is licensed according to the “PostgreSQL License” which is “a liberal Open Source license, similar to the BSD or MIT licenses” [49], [50]. As Rails natively provides a database adapter PostgreSQL, it

can be directly used with ActiveRecord, Rails' default object relational mapper. Due to its extensive use of open-source components and due to being interpreted code instead of compiled code, *Atlas* is very easy to extend.

For automated data plotting, *Atlas* uses the `cairognp` terminal on `gnuplot 5.4`. For this, no Ruby binding libraries are used. Instead, data and `gnuplot` instruction files are generated on the file system and passed as arguments to dynamically generated system calls to `gnuplot`. This creates an I/O-situation in which `gnuplot` is run in a separately spawned operating system thread which are not prone to Ruby's global interpreter lock (GIL) and therefore graph generation can easily be parallelized using Ruby's built-in threads.

4.2 Hardware, operating system and parallel computation strategy

Atlas runs on an x64 compatible personal computer with an AMD Threadripper X1950 CPU that exhibits 16 physical and 32 logical cores. It is mounted in a TR4 socket of an ASUS Prime X399-A mainboard. As RAM, two banks of DDR4-3200 CL15 modules with 8 GB each are used. The operation system, *Atlas* itself and its first-tier data storage reside on a 512 GB Samsung 960 Pro M.2 PCIe 3.0 drive which exhibits a mean time between failures of 1.5 million hours and a 4K random read/write speed of 330.000 per second. *Atlas*' second tier data storage is a RAID 1 system made up of two identical 5,900 rpm, 4 TB drives from Western Digital. Although, as a LIMS server, the computer is operating headless most of the time, a generic Nvidia Geforce GT1050 graphics card has been added for local maintenance. Furthermore, the presented *Atlas* implementation utilizes Ubuntu 18.04 LTS which is open source and which is licensed according to the GPL.

LIMS are potentially used by a large number of clients. Therefore, in order to make full use the 32 logical cores of the *Atlas* server, a stack of Apache as web server and Unicorn as application server is used. Unicorn's worker threads run in separate operating system processes which circumvents CRuby's GIL and this enables concurrent request to be handled in parallel on different CPU cores.

4.3 Primary keys, identity and labelling system

As described in Section 2.3.1, identity depends on the existence of an attribute, or a set of attributes, that is unique for each record of a given table. In SQL, this attribute or set of attributes is called "primary key". If the values of the primary keys

are unique across all tables of a database, they alone are sufficient to safely identify any record in any table. If the primary keys in different tables may assume the same set of values, only the combination of table name and primary key becomes a unique identifier for all records in a database. *Atlas* uses PostgreSQL's built-in functionality for the generation of auto-incremented sequences of integers as primary key for all tables. These, by design, cannot yield a number twice and can therefore, together with their class names, be used for identifying all entities in *Atlas*. These composite identifiers, e.g. "wear test 57" or "material 25", can also be used for labelling physical entity instances which makes it easy to find database records for physical entities and vice versa.

For reasons of maintainability, it is of course possible to manipulate the value of a field that is declared as a primary key and to also manipulate the current value of a sequence that tracks the integer sequence of a table's primary key. However, the SQL statements that are required to do so are different from those used for regular create, read, update and delete (CRUD) operations on records. While this is already enough to prevent any unintended change of ids, the application should be programmed in a way that no potentially dangerous user input can get through to the database. However, this is a difficult task by nature and even experienced developers can produce code glitches that expose a database to the execution of arbitrary - and thus potentially dangerous - code. Therefore, the database login credentials used by *Atlas* in its normal production environment correspond to a PostgreSQL user account with restricted privileges so that neither ids of existing records can be altered nor that the current value of the corresponding id sequence could be altered. Additionally, in order to prevent code glitches from making the root login available to standard users in production, the root user can only login to the PostgreSQL console off-application via a server terminal. For more information on how to grant database operation privileges to specific users and how this relates to data safety and data integrity, refer to the PostgreSQL user manual [51] or to other specialized literature [52].

4.4 Materials and material lots database

Atlas exhibits a very basic sub-database for materials which provides fields for persisting material name, material number and AISI-code, EN Ax-code, an alphanumerical "fillers" field for persisting the fillers of particle filled polymeric composites,

as well as a few fields for basic material properties: density, coefficient of linear thermal expansion, thermal conductivity and heat transfer coefficient to air. All of these material properties are dependent on state, e.g. on temperature. However, *Atlas* only provides a single numeric field for their persistence which should be used to represent the value of these properties at standard ambient temperature, humidity and atmospheric pressure.

Materials have an optional one-to-many relationship to material lots and an optional relationship to a company model which should be used to persist the manufacturer of the respective material. This field is optional since its persistence is not useful for generic materials like AISI 304 stainless steel (1.4301) which is typically manufactured by a wide range of companies without being branded individually.

Figure 5 shows the form that *Atlas* provides for creating new and editing existing materials and their attributes and the index table it provides for the materials database.

Materials

Actions:

Add new Item

ID n.n.

Name

Material no

EN Ax

AISI code

Density [g/cm³]

CLTE [10⁻⁶ K⁻¹]

Thermal conductivity [W/Km]

Heat transfer coefficient [W/Km²]

SAP

Fillers

Manufacturer

oder

Persist form?

Materials

Item index

← Previous 1 2 3 4 5 6 7 8 9 ... 25 26 Next →

ID	Material name	Material no.	EN	Manufacturer
2577	FlexiFill	(no material no)	(no en no)	no manufacturer
2576	Luvocom 70-9351	(no material no)	(no en no)	Lehmann & Voss & Co. KG
2575	T5 ICF 30 schwarz	6410	(no en no)	AKRO-PLASTIC GmbH
2574	D116-21-2	(no material no)	(no en no)	Solvay Specialty Polymers
2573	Grivory HT2C -3X LF	(no material no)	(no en no)	EMS Chemie AG
2572	Grivory HT2C -3X	(no material no)	(no en no)	EMS Chemie AG
2571	Delrin 500 NC010	(no material no)	(no en no)	DuPont

Figure 5: Form for creating new and editing existing materials (left) and Material index table (right, simplified).

4.5 Test specimens

4.5.1 Domain

Atlas has models for the following types of test specimens: block, ring, ball, disc, bushing, cylinder and plate. Table 2 shows their respective specific attributes and relationships. Additional attributes that are common to each test specimen type are material and lot, (nominal) hardness, status, custom label and several attributes for different basic roughness parameters, specifically, R_a , R_z , R_p and R_v . Of these,

“custom label” can be used to persist custom labels of individual test specimen, e.g. when test specimens have been received from external source and which carry identifiers that are either incompatible to *Atlas*’ own identification system but which need to be persisted too. For example, when external documentation relies on these labels, their persistence in *Atlas* can make external data relatable to internal test specimen identifiers.

Table 2: Test specimen classes modelled in *Atlas* and their specific attributes.

Attribute	Block	Ring	Ball	Disc	Bushing	Cylinder	Plate
width	X	X					X
gauge	X						
length					X	X	X
cross_section	X			X	X		X
height	X			X			X
inner_diameter		X		X	X		
outer_diameter		X		X	X		
diameter						X	

Test specimens do not only exist as individual entity instances but they also come in lots. Such test specimen lots consist of a set of test specimens that originate from a single production run, are made of material of the same material lot and exhibit a uniform set of attributes. Therefore, when test specimen lots can be used to represent a larger number of individual test specimens without creating an individual identity for each of them. The classical example is a lot of steel rings that have been manufactured in the same production run, e.g. surface finishing, and therefore exhibit highly uniform values for all tribologically relevant attributes, like outer diameter, material, hardness and roughness.

On an even higher level, test specimen lots belong to a test specimen class which defines geometry type, nominal values for geometry-related attributes as well as the material that the test specimens are made of. This three-tiered model delegates the definition of a test specimens attributes in a way that allows the user to choose the desired level of granularity with which a wear tests test specimen should be identified, depending on customer or project requirements. Furthermore, it enables the resolution of test results for a single material down to the level of individual material or production lots for test specimens which in turn allows for the identification of the influence of material or test specimen production on the results of friction and wear. Finally, it helps with tracking down tests that might have been

affected by errors. For example, when after a test series or project has already been concluded an error that has gone unnoticed for a while but which has eventually been identified, e.g. a ring test specimen lot with a wrong surface roughness, the exact collection of tests that has been done with this lot can be identified when either all individual rings have been related to a joint ring lot or when the tests in question have been related to this ring lot since all individual tests were considered to be equivalent and therefore no individual ring identities had been created.

4.5.2 Data modeling

Atlas uses a general class to define general attributes, properties and relationships that are the same for all types of test specimens, e.g. their belonging to (zero or more) wear tests, and then uses specialization to model type-specific behavior, see Section 2.2.3. Despite their attributes being significantly disjoint, especially their geometry-related attributes, STI is used to persist the instances of the various subclasses on the database level. The main reason for this the ease of database actions across the collection of all specimens of all types. Figure 6 shows the ERD for this model.

The three-tiered hierarchy of the test specimen model is, top to bottom, modelled by three classes: `TestSpecimenClass` (TsC), `TestSpecimenLot` (TsL) and `TestSpecimen` (Ts), each of which is related to the level below it by a one-to-many relationship. According to the domain logic, the TsL class is related to `MaterialLot` class and the TsC class is related to the `Material` model. As there is a relationship between `Material` and `MaterialLot`, a circular reference is created by this model which introduces the possibility of contradiction. However, especially with generic materials, there is not always an identifiable material lot as

the basis for the production of a domain instance of the TsL class. For such cases, it is favorable to be able to create a model TsL instance without having to provide a material lot. Instead, potential collisions are avoided by providing the user with forms that use inter-linked select fields which only allow valid combinations of references. In case that these fail, are manipulated or circumvented (e.g. by API use), model-level validity checks additionally ensure that references are void of contradictions. Next, the new Ts, TsL and TsC

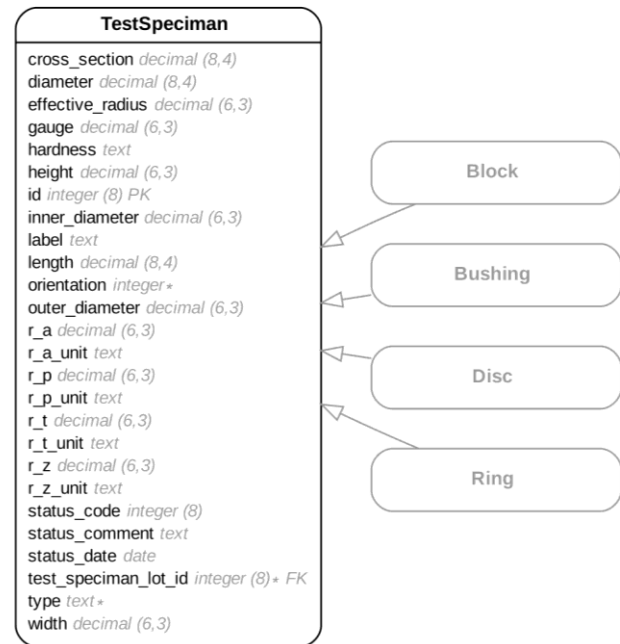


Figure 6: Bachmann-notation of *Atlas*' test specimen data model. Subclasses are printed light and are connected to their superclasses by empty-headed arrows.

classes were specialized by defining subclasses for each test specimen geometry that is supported by *Atlas*, see Section 4.5.1. Figure 7 shows the resulting ERD. For clarity, only block-type subclasses are displayed.

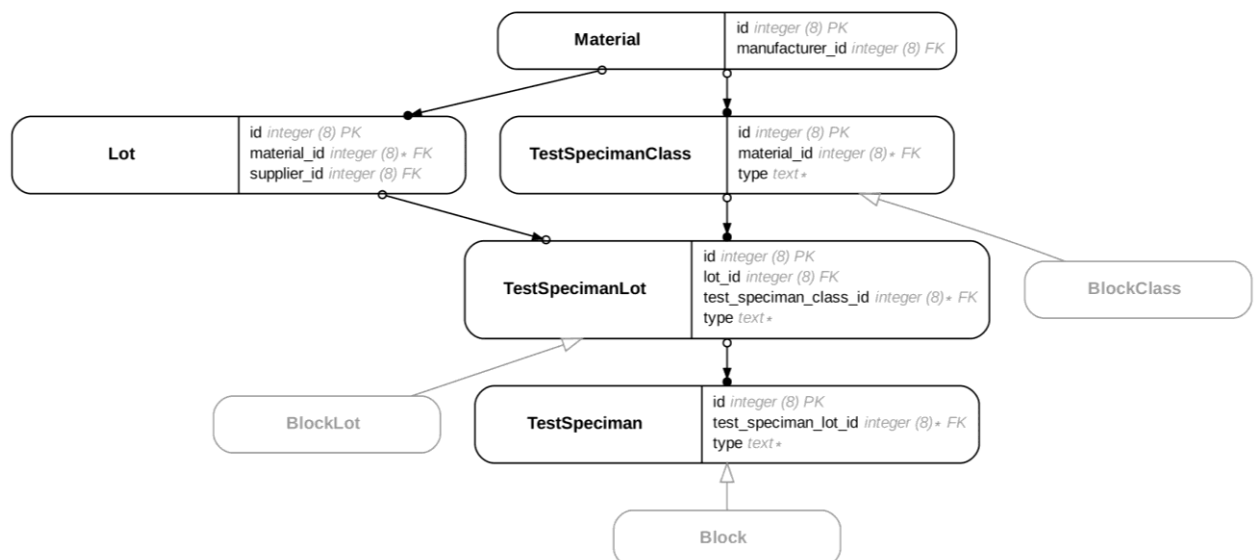


Figure 7: *Atlas*' three-level hierarchical data model for individual test specimen, test specimen lots and test specimen class. For clarity, only the block type subclasses are shown. Corresponding subclasses exist for all other supported test specimen types, see Table 2.

4.5.3 Atypical inverse property propagation

When test specimens are machined with machine tools, e.g. blocks with CNC milling machines or rings with CNC turning or grinding machines, individual variations of the actual dimensions of test specimen can be so small that the effort of measuring and recording the actual dimensions of every individual test specimen greatly outweighs the gain in accuracy. Therefore, it is feasible to not measure the dimensions of each individual specimen but to the corresponding values of the associated TsL or even of the TsC. This resembles property propagation which occurs when an aggregation object is queried for an attribute that it does not store internally but that it computes dynamically.

Therefore, the object typically does not possess an attribute of the corresponding name but instead exhibits a method of this name. However, compared to this, it is “atypical” since propagation needs to happen along a relationship between classes that are not inheriting from each other or that are components of an aggregation object. It is furthermore “inverse” as data travels from higher to lower hierarchy levels (which, due to the lack of inheritance, are solely defined by domain logic). Figure 8 shows the workflow for this mechanism. When an individual test specimen is queried for a propagatable attribute and the value that the individual test specimen has stored for this attribute is blank, the method call gets propagated to the test specimen’s TsL instance. When this has a non-blank value for the attribute in question, it is returned. When the TsL instance also does not have a non-blank value for the requested attribute, the query is propagated to the respective TsC, whose internally stored value is returned, no matter whether it is blank or not (as there is no other entity to which it could forward the request to). According to the encapsulation principle of OOP, this forwarding is not visible to the original method caller.

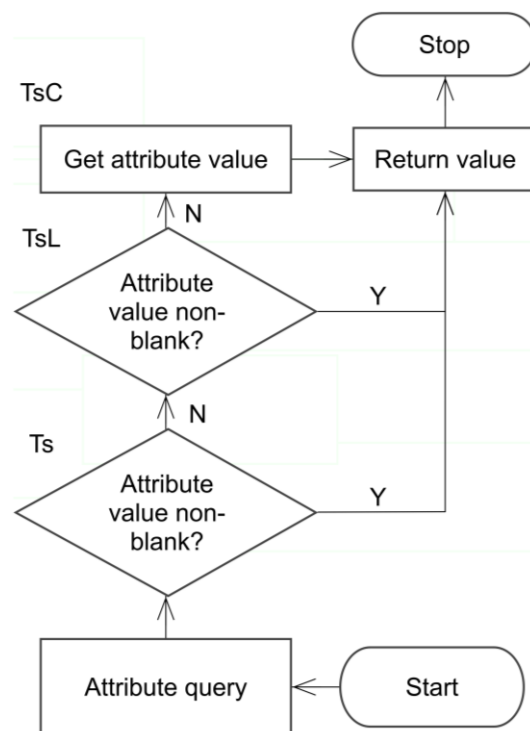


Figure 8: Atypical upstream of attribute queries and downstream of attribute data.

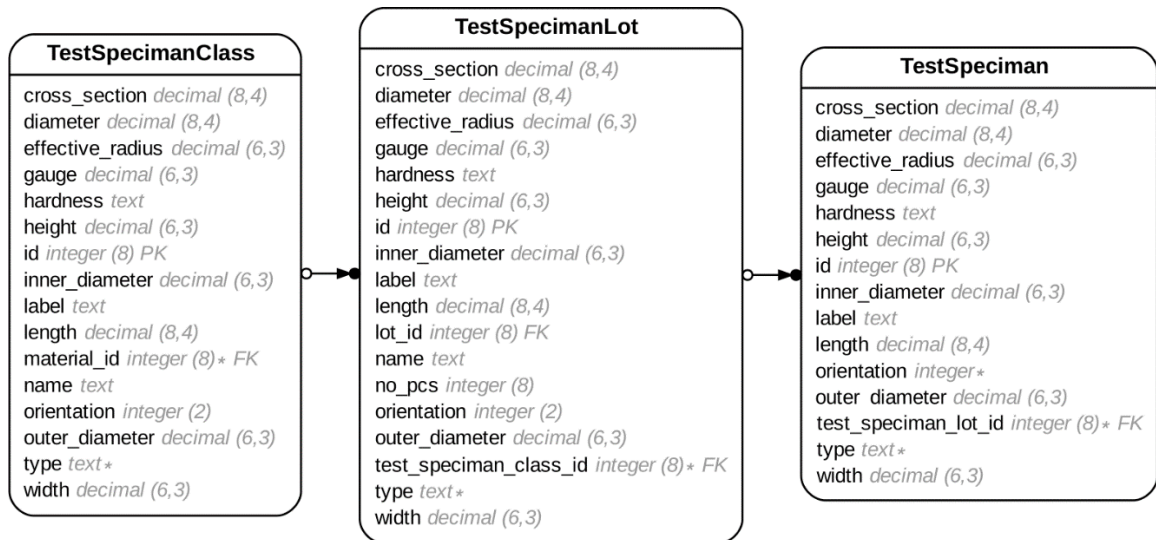


Figure 9: Test specimen data model for downstream property propagation: all propagatable attributes are present in each level of the hierarchy.

For this mechanism to work, a propagatable attribute must be defined for all three involved classes, see Figure 9. Code listing 8 shows the specific implementation of request propagation. In Rails, all classes whose instances should be persisted to the database must be subclasses of `ActiveRecord::Base` that is provided by Rails. Such subclasses automatically exhibit attribute readers, one for each database field of the underlying table. Therefore, by defining a `width` field for the test specimens table, all instances of the corresponding class, and all instances of all its subclasses, will exhibit a `width` method that return the value of the actual width field of the corresponding record, see Code listing 8. These automatically generated methods are therefore called the “attribute readers”. Analogously, a `width=` method gets defined, which will write handed-over values to the underlying database field and which is therefore called “attribute setter”.

Code listing 7: Example for the automatic generation of attribute getters and setters for subclasses of `ActiveRecord::Base`.

```

class ApplicationRecord < ActiveRecord::Base
end
class TestSpecimen < ApplicationRecord
end
class Block < TestSpecimen
  # the PostgreSQL table for test specimens
  # has a "width" field
end
b = Block.new({ width: 5})
b.width => 5
  
```

However, automatically defined attribute readers only check for the attribute of the queried instance and do not propagate the query if the internally stored value is blank. Therefore, they need to be overridden by propagating versions, see Code listing 8.

Code listing 8: Implementation of atypical property propagation for `width`.

```
# override ActiveRecord's attribute readers for width
class TestSpecimen < ApplicationRecord
  def width
    read_attribute(:width) || self.test_speciman_lot.send(:width)
  end
end
class TestSpecimenLot < ApplicationRecord
  def width
    read_attribute(:width) || self.test_speciman_class.send(:width)
  end
end
# create instances
block_class = BlockClass.new({ width: 5 })
block_lot = BlockClass.new({ test_specimen_class: block_class })
block = Block.new({ test_specimen_lot: block_lot })
# query individual test specimen for its width
block.width => 5
# check whether the result was propagated
block.propagated?(:width) => true
```

Propagation happens when `read_attribute(:width)` which reads the actual value of an object's `width` attribute from the corresponding database record, yields a blank (`nil` which represent SQL's NULLs in Ruby). In this case, the logical "or" operator (`||`) calls the alternative expression which retrieves the associated `TsL` instance and queries it for its `width` attribute. This query is handled by the `width`-instance method of the `TsL` instance which is also overridden with a propagating version of an attribute getter. If the attribute value of the `TsL` instance is also blank, the request will be forwarded to its associated `TsC` instance. Whatever the value of this `TsC` instance is for `width` will then be returned, including a potential blank value. This implementation could then be reproduced for each propagatable attribute. However, this would lead to redundant and therefore error prone code. Ruby supports "metaprogramming", i.e. the programmatical writing of code [51] [52]. For this, Ruby objects possess the `define_method` method which works like the `def` keyword (however without acting as a scope gate).

Code listing 9: Implementation of inverse property propagation.

```

module AttributePropagation
  def self.included(base)
    def base.propagates_blank(*attributes_list, to:)
      class_variable_set :@@propagated_when_blank,
        [attributes_list].flatten.map(&:to_sym)
      class_variable_set :@@propagate_blank_to, to
      attributes_list.each do |attr|
        define_method("real_#{attr}") { read_attribute attr }
        define_method("real_#{attr}=") { |new| write_attribute attr, new }
        define_method(attr) do
          self.send("real_#{attr}") || propagate_blank_to.send(attr)
        end
      end
    end
  end
end

def propagated_when_blank
  self.class.class_variable_get :@@propagated_when_blank
end

def propagated_when_blank?(attr)
  propagated_when_blank.include? attr.to_sym
end

def propagate_blank_to
  self.send self.class.class_variable_get :@@propagate_blank_to
end

def propagated?(attr)
  return false unless propagated_when_blank?(attr)
  self.send("real_#{attr}") .blank?
end
end

```

Code listing 9 shows the implementation of the `AttributePropagation` module. When included into a class, represented by the first positional argument of `Module.included`, it provides this class with the new class method `propagates_blank` which accepts a list of attribute names that should be propagated and the name of an association that attribute queries for blank attributes should be propagated to. Code listing 10 shows how the new class method `propagates_blank` is used to declare the propagation of the attributes `width`, `gauge` and `height` when they are blank. In Ruby, “class variables” are actually “class hierarchy variables”, which means they are accessible from all classes of an inheritance hierarchy. Therefore, it is sufficient to include the module into the base class of the three levels of *Atlas* test specimen taxonomy: `TestSpecimenClass`, `TestSpecimenLot` and `TestSpecimen`.

Code listing 10: Property propagation using `AttributePropagation`.

```

class TestSpecimen < ApplicationRecord
  include AttributePropagation
  propagates_blank :width, :gauge, :height, to: :test_specimen_lot
end
class TestSpecimenLot < ApplicationRecord
  include AttributePropagation
  propagates_blank :width, :gauge, :height, to: :test_specimen_class
end

```

When rendering HTML views, the `propagated?`-method can be used to visualize whether individual attribute values have been propagated. Figure 11 shows a test specimen index table that visually indicates the different sources of displayed attributes by using different type set styles.

ID	TsC ID	TsL ID	Individual label	Material name, Material lot name	Geometry	Hardness	Width [mm]	Gauge [mm]	Height [mm]	Outer diameter [mm]	Inner diameter [mm]	Length [mm]	Cross section [mm ²]
35516	917	1028	<i>(none)</i>	PEEK 450G Arkema, no lot	Block		4.0	4.0	2.0				16.0
35515	917	1028	<i>(none)</i>	PEEK 450G Arkema, no lot	Block		4.0	4.0	2.0				16.0
35514	917	1028	<i>(none)</i>	PEEK 450G Arkema, no lot	Block		4.0	4.0	2.0				16.0
35513	917	1028	<i>(none)</i>	PEEK 450G Arkema, no lot	Block		4.0	4.0	2.0				16.0
35512	928	1039	<i>(none)</i>	EP-GR17 (RGC39A), no lot	Block		4.0	4.0	2.0				16.0
35511	928	1039	<i>(none)</i>	EP-GR17 (RGC39A), no lot	Block		4.0	4.0	2.0				16.0
35510	928	1039	<i>(none)</i>	EP-GR17 (RGC39A), no lot	Block		4.0	4.0	2.0				16.0
35509	928	1039	<i>(none)</i>	EP-GR17 (RGC39A), no lot	Block		4.0	4.0	2.0				16.0

Figure 10: Test specimen index with different type setting for propagated (grey, italic) and non-propagated (upper face, bold) attributes of individual test specimen (here: width, gauge, height and cross section of blocks).

4.5.4 Test specimen labelling and archiving

Each of the three levels of the test specimen model has a `custom_label` attribute in order for satisfy the domain requirement of persisting custom test specimen labels, e.g. from an external source. However, external labels are optional and potentially non-unique and are therefore only useful for relating internal identification to external identification.

For actual identification, *Atlas* uses PostgreSQL sequences to assigns sequential integers to the id fields all records. In combination with the respective class name, this forms a unique identifier for all entities of a given *Atlas* server instance. For test specimens, there is even only one sequence for each hierarchy level as all subclasses of a given hierarchy level are persisted with STI. Therefore, the name of the hierarchy's superclass can be used to form a unique human-readable identifier. In order to enable a compact representation, "individual test specimen" is abbreviated by "TS", "test specimen lot" by "TSL" and test specimen class by

“TSC”. For an even more compact version, this can be further reduced to “S”, “L” and “C”.

Code listing 11 show the implementation of this convention in the form of a module and its exemplary integration into the superclasses of the hierarchy level for individual test specimens.

Code listing 11: Implementation of the `LabelLable` module and its use in test specimen-related superclasses.

```
module TestSpecimenLabel
  def self.included(base)
    def id_label(variant: :normal)
      uppercase_chars = self.class.base_class.to_s.scan(/[A-Z]/)
      prefix = case variant.to_sym
                when :normal then uppercase_chars.join
                when :compact then uppercase_chars.last
              end
      "#{prefix} #{id}"
    end
  end
end

class TestSpecimen < ApplicationRecord
  include TestSpecimenLabel
end

# load existing record
block = Block.last
block.id_label
=> TS 34589
```

The labels that are generated by this code then can be used in the laboratory for sorting and archiving test specimens. Individual test specimens can be archived in sequence, e.g. in ascending order of their id. As all subclasses of a given test specimen superclass use the same id counter, all individual test specimens can be easily stored in a joint storage which enables the retrieval of a range of mixed-type specimens in one work step. Test specimens that are collectively label by their lot-based label (e.g. “TSL 357”), can be grouped together in suitably sized storage containers and also archived in ascending order. It is recommended to use three separate storage locations, one for each hierarchy level, in order to prevent id collisions and in order to be able to operate each storage in append mode only.

4.5.5 GUI elements for workflow support

Given the mandatory role of test specimen lots in their relationship to individual test specimen, any test specimen lot that a new test specimen ought to be related to must exist before any individual test specimen can be created. The same is true for test specimen classes and lots. Therefore, the natural workflow is top-down: create a new or select an existing TsC, then create a new or select an existing Tsl and then create any new individual test specimen.

The figure displays three screenshots of the Atlas GUI forms for creating test specimen entities:

- Test Speciman Classes:** This form includes fields for ID (n.n.), Class type (Ring), Material (PEEK), and Material ID or name (2264: Victrex PEEK 150FC30). It also has fields for Name, Hardness, Width (28 mm), Inner diameter (50 mm), Outer diameter (60 mm), and surface roughness parameters (Ra: 0.1 μm, Rz: 1.7 μm, Rp, Rt). Orientation is set to Parallel. There are buttons for Code (OK), Status, Comment, Date, and a checkbox for 'Persist form?'.
- Test Speciman Lots:** This form includes fields for ID (n.n.), Test Speciman Class (854: Victrex PEEK 150FC30), Geometry (Ring), and Material lot (Lot 1250). It has fields for Name, Hardness, Width, Gauge, Inner diameter, Outer diameter, Cross section (mm²), Diameter, Length, Effective radius, and Height. Surface roughness parameters (Ra, Rz, Rp, Rt) are shown as dropdown menus. Orientation is 'as per test specimen class'. There are buttons for Code (OK), Status, Comment, Date, and a checkbox for 'Persist form?'.
- Test Specimen:** This form includes fields for ID (n.n.), Geometry (Ring), Material (Select an Option), and Test specimen lot (- select material -). It has fields for Custom label, Hardness, Width, Inner diameter, Outer diameter, and surface roughness parameters (Ra, Rz, Rp, Rt) as dropdown menus. Orientation is 'as per test specimen lot/class'. There are buttons for Code (OK), Status, Comment, Date, a 'Create 1 identical specimen' button, and a checkbox for 'Persist form?'.

Figure 11: Forms for creating new test specimen in *Atlas*.

Figure 11 shows the three forms that *Atlas* provides for these operations. For attributes that establish relationships, e.g. between a TsC instance and a material instance, automatically populated select fields are provided. Thus, the user can only select entities that are valid choices for the underlying foreign key restrictions. For all other attributes, input fields are provided.

While the type-specific attributes of each subclass must be specified when creating new test specimen class instances, this is optional when defining test specimen lots or individual test specimens. When non-blank values are provided for attributes that are propagated, see Section 4.5.3, attribute propagation will be interrupted and instead these values will be returned when the corresponding instance is queried for the attribute in question.

4.6 Tribometers

4.6.1 Domain

While most tribometers can only be equipped with one set of test specimens, some have been designed to host multiple sets. In this case, they typically exhibit a corresponding number of individual test specimen mounts, see Figure 12.

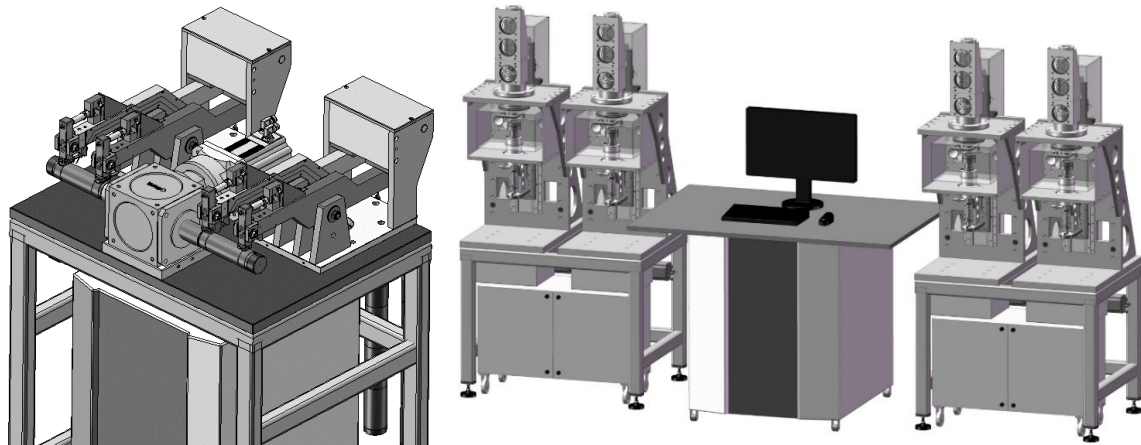


Figure 12: Left: block on ring-tribometer with four parallelized sets of block/ring-pairs. Rings are mounted to one common shaft that is driven by a drive. Right: pin on disc-tribometer with four parallelized sets of pin/disc-pairs.

4.6.2 Data model

As tribometers exhibit a variable number of test positions, each of which can take an individual test specimen pair (or more general, a tuple), tribometers are modelled as composite objects with test positions as their components. This one-to-many association is mandatory on both sides, see Code listing 12 and Figure 13.

Code listing 12: Association between tribometer and testing position(s).

```
class Tribometer
  has_many :positions
  validates :positions,
            length: { minimum: 1 }
end

class Position
  belongs_to :tribometer
  validates_presence_of :tribometer
end
```

Trivially a tribometer has exactly one test position and the association collection consists of only one object. Still, the one-to-many association allows the digital representation of tribometers with multiple test positions. Being a composite object, there are attributes that belong to a tribometer, e.g. name, inventory number, service status or the IP address of its control computer which are implemented as attributes of the tribometer class itself.

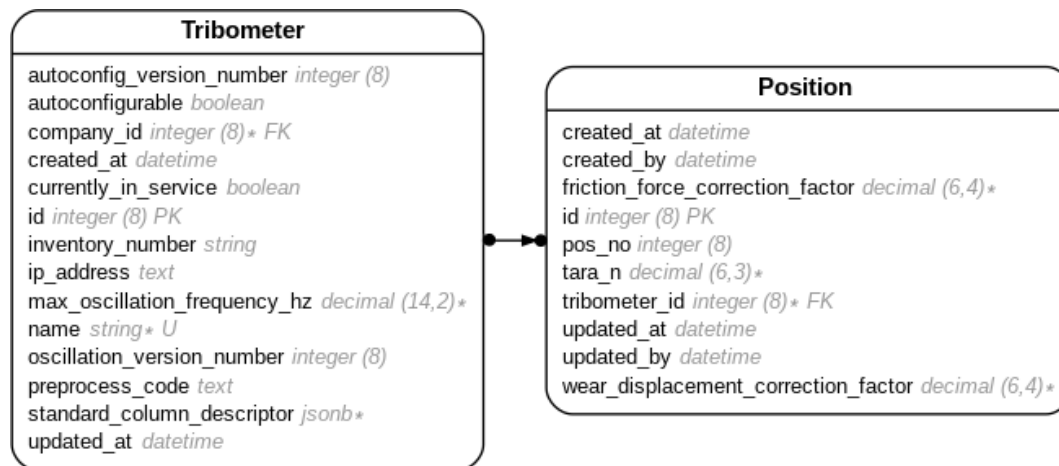


Figure 13: Modelling of the relation of wear tests and tribometers by (testing) positions.

On the other hand, there are attributes whose values are specific for each individual position, e.g. geometry-related correction factors for friction force and height loss signals or a human-readable, tribometer-specific numbering of the test positions, e.g. one-to-many. The exact number of test positions of a tribometer is an example for a propagated attribute, see Section 2.2.5, with the propagation algorithm being as simple as counting the number of associated objects, see Code listing 13.

Code listing 13: A tribometer’s number of test positions is a propagated attribute.

```

class Tribometer
  def number_of_test_positions
    positions.size
  end
end

```

4.7 Wear Tests

4.7.1 Domain

Wear tests belong to the central entities of a tribological laboratory. Important attributes of wear tests are the test parameters that have to be applied, e.g. sliding speed, oscillation frequency, test duration, loading etc. A wear test is called “non-segmented” if the same set of test parameters is applied over the whole test duration. If one or more test parameters need to change during the course of a test, the test is called “segmented”. Whether a test can be executed in segments or not is

a property of the corresponding tribometer. When a test is segmented, the collection of test parameters must be subdivided into a set that is subject to segment-wise alteration (segment-specific parameters) and into a set of parameters that are not subject to alteration within segments but which are constant during all segments (test-specific parameters). Again, the total set of available parameters and their distribution into these two sets is specific to the respective tribometer and depends on its mechanical design, its electronics and its programming. When a tribometer uses static weights to apply the normal load, normal load is a test-specific parameter. When it used computer-controlled devices like hydraulics, pneumatics or electromechanical actuation, it is a segment-specific parameter. Sliding speed has been set for decades using a potentiometer which made it a test-specific parameter. With modern computer-controlled drives, it has become a segment-specific parameter. A typical example for a segmented wear test is the ASTM G137 block on ring-test which requires the measurement of the static coefficient of friction, followed by a period of uniform, unidirectional sliding, followed by another measurement of the static coefficient of friction [53]. Furthermore, wear tests have important metadata attributes like time of creation, modification and start of test. Metadata contains valuable information: timestamps for the start of test can be used to verify that a test has been done in a time interval for which the tribometer's sensors have been calibrated or to combine recorded test data with data from disparate devices, like camera equipment (see Section 5) or from climate control devices. Furthermore, data about the validity of a test's primary data and its results are of interest, e.g. a test can have invalid real time wear loss data because of a malfunctioning of the wear tracking sensor while the data recorded by other sensors is perfectly valid.

Wear tests are not isolated entities but have associations to many other entities in a tribology laboratory. Amongst them are the tribometer on which the test should be (or was) done, the set of test specimen on which the test was performed, the person who conducted the test (the operator). Furthermore, the test standard that was followed the lubricant that was used, the test set they belong to and finally the project that they belong to. Finally, every wear test must be related to its results, which mainly consist of the collected primary data during the test, but also of additional results, like analytical results which can be qualitative, e.g. microscopic images taken of the test specimen before and/or after the test, or numeric results like

post-test surface roughness. Additionally, wear tests can be related to arbitrary information in the form of generic entities like document or comment. Typical examples include photographs of the test setup or comments of the operator on special occurrences or of the project manager on specific details of the interpretation of the test's results.

4.7.2 Data model

The logic data model for wear tests, like the one for test specimens, uses the design pattern of specialization, see Section 2.2.3. Starting from one `WearTest` superclass which includes all the general attributes behavior and relations of wear tests, a set of subclasses is defined, see Figure 14.

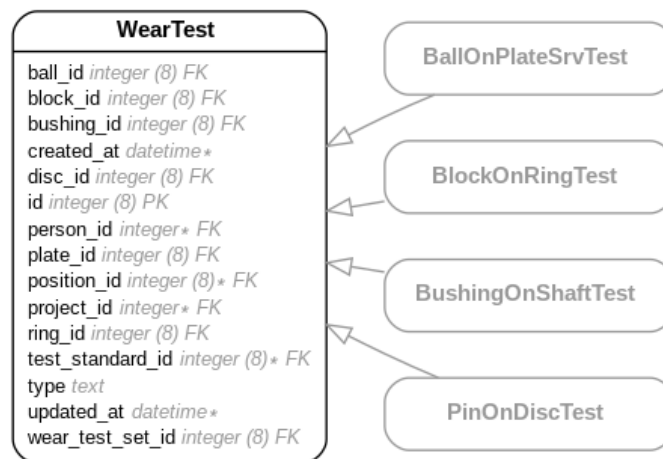


Figure 14: ERD of *Atlas*' data model for wear tests

Each subclass represents a specific type of wear test, e.g. block on ring-tests or pin on disc-tests. By default, each of the subclasses inherits the attributes, relations and methods of the `WearTest` superclass and can then implement the specific behavior of the test it represents by overriding the corresponding methods. On the level of the physical data model, *Atlas* persists wear test instances using single table inheritance, see Section 2.5. This allows wear test segments to relate to any subclass of the wear test model using a standard binary relationship to the wear test superclass.

Due to the segmentation of wear tests into individual segments, wear tests are composition objects with segments as their components. As per the domain rules the number of segments is at least one, the cardinality of this composition is one-to-many. Many attributes that one might naturally consider to be attributes of a wear test instance actually are attributes of the individual wear test segments.

Therefore, test parameters like sliding speed, duration or temperature are not attributes of the wear test to which they belong to but of the individual wear test segments. While within a segment all test parameters are constant, one parameter, none or all may change when one segment ends and the next segment starts. Consequently, the same applies to “test results” like COF, linear wear rate, counter body temperature et cetera which are actually “segment results” in *Atlas*.

Figure 15 shows the data model of wear tests and their segments and Code listing 14 shows the establishing of this one-to-many composition on the code level.

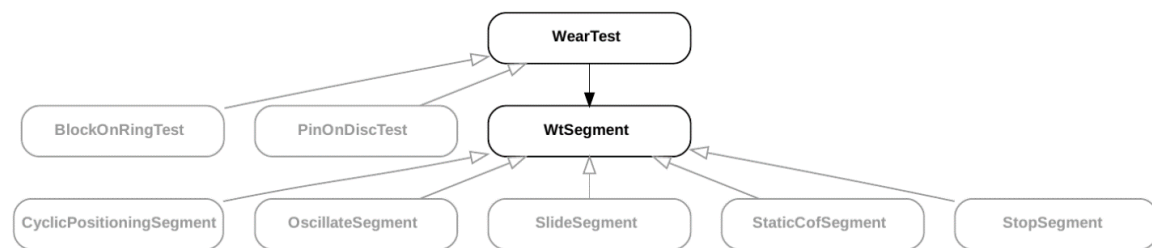


Figure 15: ERD of the wear test and wear test segment models including their respective subclasses

Code listing 14: Declaration of the relationship between tests and test segments.

```

Class WearTest < ApplicationRecord
  has_many :wt_segments
  validates :wt_segments,
            length: { minimum: 1 }
end

class WtSegment < ApplicationRecord
  belongs_to :wear_test
  validates_presence_of :wear_test
end
  
```

Additional complexity arises from the different operation modes of tribometers which typically serve the purpose of measuring different quantities. For example, it is a requirement of the ASTM G137 standard to measure the static COF before and after each sliding wear test. This mandates the definition of such a test in at least three individual segments: static COF measurement, uniform sliding, static COF measurement. Each of these segments requires a set of test parameters of its own. Therefore, *Atlas* uses specialization to create specific subclasses that represent all relevant types of wear test segments: sliding, oscillation, cyclic positioning, measuring of the static COF and stopping. On the level of physical data storage STI is the preferred choice, as it enables the establishment of the association between all subclasses of wear tests and wear test segments by one single declaration on the level of the two superclasses which then gets inherited to both subclass hierarchies.

Overall, wear test segments exhibit 264 attributes that are distributed across six categories: test parameters, evaluation results, validity data, evaluation parameters, metadata and miscellaneous. Each of these may contain attributes that are common to all subclasses and subclass-specific attributes. Due to their different purpose and different execution during tribometer operation, type-specific attributes are most common in the group of test parameters, see Table 3.

Table 3: Wear test segment subclasses and their domain-specific attributes.

Attribute	Slide	Cyclic Positioning	Oscillate	Stop	Static Cof
sliding_speed	x				
oscillate_speed_forwards			x		
oscillate_speed_backwards			x		
sliding_distance	x		x		
oscillation_frequency			x		
stroke			x		
duration				x	
static_cof_rate					x
static_cof_latency					x
positioning_type		x			
positioning_cycles		x			
positioning_duration		x			

While a segment that represents uniform sliding over a given period of time requires the total sliding distance, the sliding direction and the absolute of the sliding speed, an oscillation segment requires oscillation frequency, stroke and total sliding distance. As sliding speed, total sliding distance and duration are not independent of each other, *Atlas* assumes sliding speed and total sliding distance to be independent and will, when asked, compute the duration from these two parameters. However, there are models for which sliding distance is no suitable test parameter, e.g. stop segments. For such cases, *Atlas* provides an explicit duration attribute, whose value is then returned directly when queried for. Ruby's natural support for the uniform access principle, see Section 2.2.7, makes this different implementation (return value of a method vs. attribute) very easy because method calls and attribute queries use exactly the same syntax (if the method call does not require the supplement of attributes), see Code listing 15.

Test parameter attributes that are common to all segment subclasses include, but are not limited to, ambient temperature, set specimen temperature (e.g. when heating the steel ring in a block-on-ring test), temperature control mode, loading, maximum eligible wear, initial lubricant amount, lubricant feed rate, as well as ambient

temperature and humidity. Furthermore, there are “non-tribological” parameters that are needed for parameterizing tribometer operation and data acquisition, like logging frequency and averaging time as well as flags for taring sensors at the beginning of the segment and a corresponding pre-tare latency.

Code listing 15: Implementation of the duration of a wear test segment as a derived attribute using UAP (handling of units omitted).

```

Class SlideSegment < WtSegment
  def duration
    sliding_distance / sliding_speed
  end
end

class StopSegment < WtSegment
  # without an explicit definition
  # ActiveRecord's default attribute
  # reader will be used
end

segment_1 = SlideSegment.new({ sliding_speed: 1, sliding_distance: 21600 })
segment_1.duration
=> 21600

segment_2 = StopSegment.new({ duration: 3600 })
segment_2.duration
=> 3600

```

Metadata includes timestamps for the datetimes of creation and (last) update of the record as well as whether the segment has already been executed (“done”), whether a measurement data file has already been uploaded and whether it has already been evaluated as well as a timestamp for the beginning of segment execution (`started_at`). While the presence of a timestamp for `started_at` already implies that a segment has been done, there may be rare cases when it is known that a segment has been executed, but the timestamp for the start of execution is unknown. For such cases, *Atlas* overrides the default attribute reader for `done` that will return true even if the actual value for `done` is false (which is this field’s default value) but a timestamp is present for `started_at`.

Another important meta information is the order in which wear test segments should be executed. The id of a segment is no useful order criterion since it cannot be modified after creation. Furthermore, programmatic segment creation via mass assignments might not always yield the intended order. Therefore, wear test segments exhibit a `segment_index` attribute which is used for ordering segments of the same wear test.

4.7.3 Attributes and Associations

Wear tests have a wide range of attributes and relationships. First, they need to be related to the test specimens that should be used. While, for example, a block on

ring test typically utilizes a block and a ring, a pin on disc test will use a pin and a disc. Typically, the number and types of test specimen is highly individual to the specific type of the test. This variability has been modeled by specialization, i.e. each type of test has been modeled as a subclass of the generic wear test class. On the database level, single table inheritance has been chosen as mechanisms for persisting the instances of the various subclasses. With the aim of ensuring that test specimens can be identified for any wear test, business logic mandates that the relationships between wear tests and their test specimens have been implemented as mandatory. However, the combination of representing specialization by STI on the data base level with the hierarchical model for test specimen (class, lot, individual) poses a special challenge for the technical implementation of the relationship between wear tests and test specimens. Specifically, it was not possible to represent the requirements for the presence of declared test specimens using foreign key restraints on the data base level. While data base-level triggers and procedures would have been a viable alternative for this, polymorphic associations and application-level constraint checking have been implemented due to their ease of use, see Code listing 16. While this basically makes *Atlas* prone to read/write anomalies, this has been found not to be a serious problem for low- to medium traffic Rails applications (like *Atlas*), even when they utilize Unicorn for a multi-process approach to concurrent request handling (as *Atlas* does) [54]. As a consequence of these design decisions, the wear test table exhibits one type and one id field for each of the possible test specimen geometries (block, bushing, disc, plate, ball, cylinder and ring). By declaring these relations to be polymorphic, Rails automatically uses the value of the corresponding type field to identify the target table and the value of the id field to identify the correct record from the target table.

Code listing 16: Declaration of the relationship between block on ring- and ball on plate-tests and their respective test specimens using class methods provided by ActiveRecord.

```
Class BlockOnRingTest < WearTest      class BallOnPlateSrvTest < WearTest
  belongs_to :block,                  belongs_to :ball,
    polymorphic: true                 polymorphic: true
  belongs_to :ring,                  belongs_to :plate,
    polymorphic: true                 polymorphic: true
  validates_presence_of :block        validates_presence_of :ball
  validates_presence_of :ring         validates_presence_of :plate
end                                    end
```

In order to enable Rails to resolve this relation the other way round, a complimentary declaration must be made on each of the three levels of the test specimen hierarchy, see Code listing 17. These declarations are then inherited to the individual subclasses within each hierarchy level.

Code listing 17: Declaration of the inverse relation between block on ring wear tests and blocks in the `Block`, `BlockLot`, `BlockClass` classes.

```

Class TestSpecimen      class TestSpecimenLot      class TestSpecimenClass
  has_many :wear_tests,  has_many :wear_tests,  has_many :wear_tests,
    as: :ring            as: :ring              as: :ring
  has_many :wear_tests,  has_many :wear_tests,  has_many :wear_tests,
    as: :block           as: :block             as: :block
  has_many :wear_tests,  has_many :wear_tests,  has_many :wear_tests,
    as: :disc            as: :disc              as: :disc
end                      end                      end

```

Figure 16 shows *Atlas*' data model that results from using a three-tiered test specimen hierarchy in combination with polymorphic associations for relating a block on ring test to its "ring" test specimen. Though omitted, for the sake of clarity, the same is true for the relation of block on ring tests to their respective rings as well as for all other test type/test specimen combinations in *Atlas*.

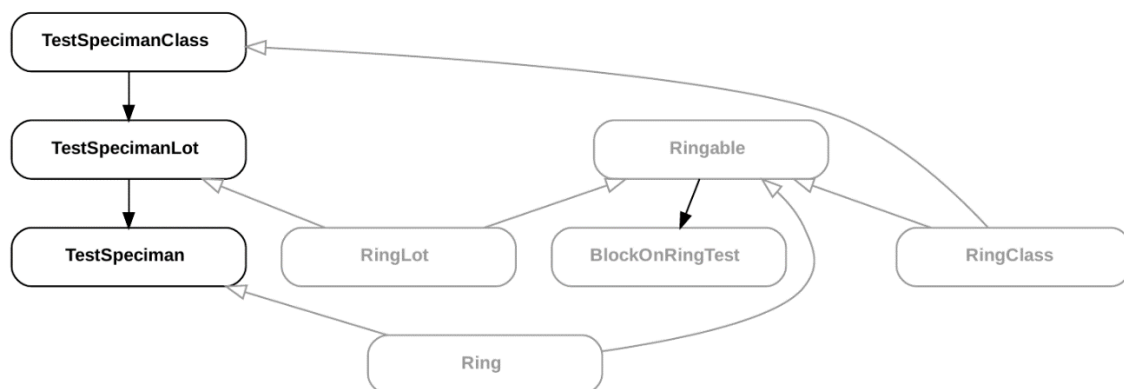


Figure 16: Polymorphic association between `BlockOnRingTest` and `Ring`. `Ringable` is the name of the polymorphic association between any of the ring-type subclasses of *Atlas*' main test specimen classes and subclasses of `WearTest`.

The ability of associating any specimen hierarchy element to any subclass of the wear test superclass enables users of *Atlas* to specify the degree of "specimen identity" they want to relate their test results to. For example, steel rings are ground to their final surface in a batch process. Due to the high reproducibility of today's machine aided grinding, such production batches typically yield ring lots with very little inter-specimen scattering of attributes, e.g. of surface roughness parameters.

Therefore, rings are typically related to block on ring wear tests only via their test specimen lot and not per individual test specimen ids. On the other hand, if required for detailed analytics, the relation can still be based on individual rings, if the user opts to do so.

In *Atlas*, tribometers are related with a set of testing positions. Wear tests are therefore not directly associated to tribometers, but to one of the positions that are associated to the respective tribometer, see Figure 17.

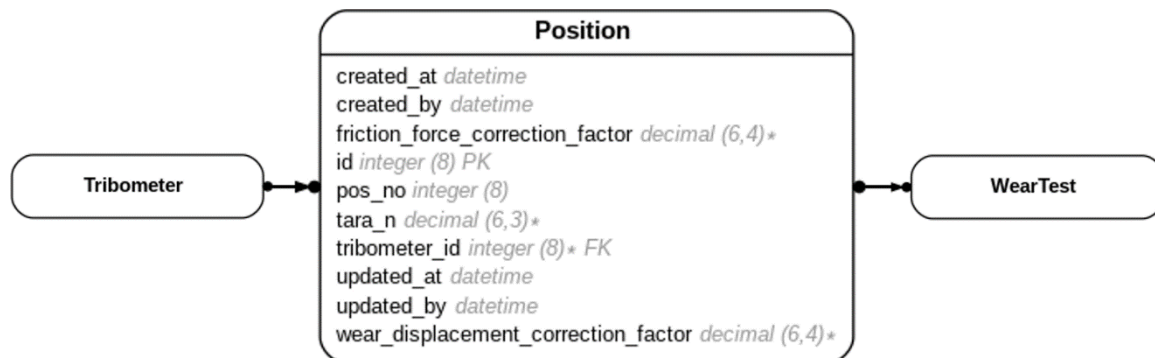


Figure 17: Relationship between wear tests, (testing) positions and tribometers.

By adding testing position-specific data to this relation, this data is easily accessible when processing the recorded primary test data later, e.g. by applying leverage ratio of a lever on displacement or force transducer data. In a more advanced data model, which is beyond the scope of this work, this design might be upgraded even more, e.g. by adding relations between the testing positions and measuring instruments that are installed at that respective position. Such an extension would establish a trace from aggregated results through recorded data to calibration certificates of sensors. Furthermore, when adding validity periods for such certificates, an automatic classification of wear tests as “done with validly calibrated sensor equipment” could be implemented.

In addition to their relationships to tribometers (through the position relationship) and test specimens (through which they are related to a material), wear tests and their segments are related to the lubricant model, the test standard model, the person model (in the role of an operator), to the graph model to the test set model and through this to the project model.

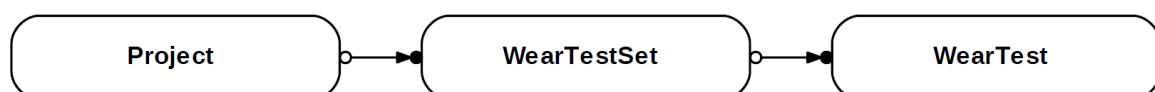


Figure 18: Relationships between projects, wear test sets and wear tests.

4.7.4 Creation of wear tests

The process of creating a wear test includes the creation of a new instance of the corresponding `WearTest` subclass, the assignment of values to all relevant attributes of the new object and the assignment of associated object instances to relationships. As described in Section 4.7.3, even in their simplest form of single-segment tests, wear tests and their segments consist of more than 100 individual pieces of information. Many of them are critical to the successful conduction and evaluation of a wear test. A good example for this is the combination of averaging time and logging frequency. When not set to (typically very narrow) correct ranges of values, which are possibly even interdependent, the tribometer control software might be perfectly able to execute the test, but the resultant measurement data file might be useless in terms of friction force data. The issue with this large number of parameters and their definition is that humans have error rates for elementary tasks that makes any process that relies on a human operator to not make an error in a high number of critical operations highly likely exhibit at least one such critical error with a non-neglectable probability. For the lack of reasonable error estimates for elementary processes in a tribology laboratory in general and for the process of entering a large amount of data into a LIMS.

Table 4: Estimates for human error rates of elementary processes, reproduced from [55] with modification.

Process	Error rate [%]
Passive inspection (general walkaround)	10.0
Inspector fails to recognize initial error by operator	10.0
Install O-Ring – improperly install	6.67
Read instructions – make procedural error	6.45
Carry out Plant Policy – no check on operator	5.00
Simple arithmetic error (without redoing calculation on separate paper)	3.00
Technician “seeing” an out-of-calibration instrument as “in tolerance”	2.00
Adjust mechanical linkage – improper adjustment	1.67
Read gauge incorrectly	0.50
Tighten nuts and bolts – not tightened	0.48
General error of omission for items imbedded in a procedure	0.30
General errors of commission, e.g. misread label and selected wrong switch	0.30
Select wrong controls	0.30
Assemble connector – bent pins	0.15
Error of omission – 10-item check off list	0.10
Selection of switch dissimilar in shape or location to desired switch	0.10
Install nuts and bolts – not installed	0.06

Table 4 shows estimates for the error rates a selection of elementary technical processes conducted by trained human operators. Although mainly determined for processes in chemical and power plants, the presented selection of processes is assumed to be relevant for tribology laboratories as well. From this, it becomes obvious that creating new wear tests from scratch, i.e. not by cloning existing tests, requires computerized support that ensures that all required data is entered, that relationships are formed only to existing and valid objects (i.e. that referential integrity is ensured and maintained, see Section 2.3.3) and that whatever data that can be auto-completed is so. For this, *Atlas* includes a guided process through a series of HTML forms that are supported by code- and database-level validation on the server side. The user starts with a basic form in which he enters general data on the projects to which the newly created tests should belong to, which tribometer ought to be used to do the tests, what general standard to follow and whether to relate the new tests to an existing wear test set or to create a new one, see Figure 19. On form submission the browser sends this information to the server using an asynchronous request. The server then uses the provided tribometer id and finds the number of test positions that are related to it. For each test position, it renders a set of HTML elements that includes inputs about the test respective test specimens for the new test. As the relationship between wear tests and their test specimens is polymorphic, two pieces of information must be entered for each test specimen: its class name and its id. This provides the user with a very high degree of granularity from which he can choose the correct level for depending on domain requirements. The server sends the rendered HTML back to the browser which then inserts it into the web page. Figure 21 shows the result of this process for the four-position block-on-ring tribometer shown on the left side of Figure 13. Next, the user enters the data that identify the test specimens that he wants to use in the new tests. By default, *Atlas* assumes that when individual test specimens are used, they will exhibit sequential ids. Therefore, any number that is entered into the id field of a test specimen on the first position, the entered number will be forwarded to the respective fields of all other positions, incrementing the id accordingly so that a gap-free sequence is created. Correspondingly, when a test specimen lot subclass is selected and an id is entered, *Atlas* will assume that the same TSL is to be used on all positions and will forward the entered id without incrementing it.

Wear tests Materials Test specimen Projects Laboratory Administration Logged in as Andreas Gebhard Log out Atlas4, Vers. 3.5.0

Create new wear test(s) **fill in general data**

General data

Project: 501: 2017-2-27 Graf 2017 (Ensinger GmbH)
 Tribometer: Atlas TT3
 Operator: Andreas Gebhard
 Test standard: - select -
 Make this test(s) a member of: - select -

Wear test set: Block on ring

Test specimen options

Autocomplete IDs when ID of lowest used position number gets changed
 Autoadjust ID increments to used number of positions whenever a position is (de)selected for use

Sliding Oscillation Sliding Duration Cyclic
 Index Pass Type Loading speed frequency Stroke distance [hh:mm:ss] positions Lubricant q₁ q₂ temp humidity wear rate [1/min] latency [s] F/W/D Tara mode Counter body Eval Applied
 Add segment(s)

Create test(s)

Figure 19: Form-based creation of wear tests for a four-position tribometer, step 1: general data.

Wear tests Materials Test specimen Projects Laboratory Administration Logged in as Andreas Gebhard Log out Atlas4, Vers. 3.5.0

Create new wear test(s)

General data

Project: 501: 2017-2-27 Graf 2017 (Ensinger GmbH)
 Tribometer: Atlas TT3
 Operator: Andreas Gebhard
 Test standard: - select -
 Make this test(s) a member of: a new set

Test specimen options

Autocomplete IDs when ID of lowest used position number gets changed
 Autoadjust ID increments to used number of positions whenever a position is (de)selected for use

Test specimen

	Position 1	Position 2	Position 3	Position 4
Block	Block	Block	Block	Block
Ring	RingLot	RingLot	RingLot	RingLot
Angle	90°	90°	90°	90°

Wear test segments

Sliding Oscillation Sliding Duration Cyclic
 Index Pass Type Loading speed frequency Stroke distance [hh:mm:ss] positions Lubricant q₁ q₂ temp humidity wear rate [1/min] latency [s] F/W/D Tara mode Counter body Eval Applied
 Add segment(s)

selecting a test standard will load an appropriate form partial for entering test specimen ids

Create test(s)

Figure 20: Form-based creation of wear tests for a four-position tribometer, step 2: test specimen data.

Test specimen

	Position 1	Position 2	Position 3	Position 4
Block	Block 29953	Block 29954	Block 29955	Block 29956
Ring	RingLot 738	RingLot 738	RingLot 738	RingLot 738
Angle	90°	90°	90°	90°

loading information snippets for validation

Figure 21: Form-based creation of wear tests for a four-position tribometer, step 3: loading information snippets for validation.

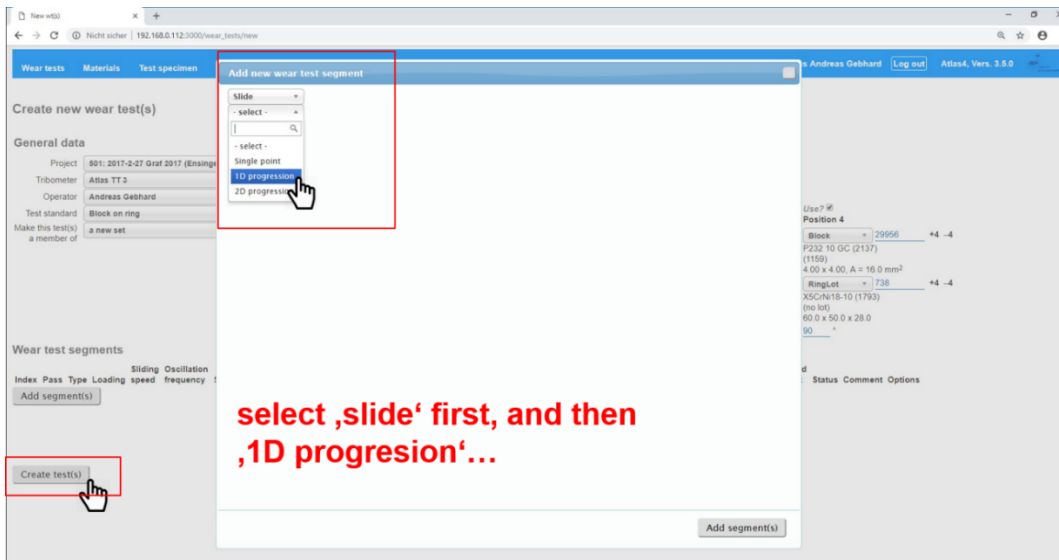


Figure 22: Form-based creation of wear tests for a four-position tribometer, step 4.1: define type and number of segments to add.

Add new wear test segment

Slide

1D progres...

with steps in

Temperat...

Sliding speed

m/s

Loading

MPa

Eval every

Averaging time [s]

Log interval [s]

Initial stage [s]

Max. wear [μm]

Segment	Mode	ϑ °C	s m	t hh:mm:ss.ms	Averaging time [s]	Log interval [s]	Initial stage [s]	Max. wear [μm]
1	Heat/cool	50	28800	4	12	2.866	1.0	300
2	Heat/cool	75	28800	4	12	2.866	1.0	80
3	Heat/cool	100	28800	4	12	2.866	1.0	80
4	Heat/cool	125	28800	4	12	2.866	1.0	80
5	Heat/cool	150	28800	4	12	2.866	1.0	80
6	Heat/cool	175	28800	4	12	2.866	1.0	80

Humidity

%

Lubricant

Dry sliding

Wear limit (per segment)

μm

Figure 23: Form-based creation of wear tests for a four-position tribometer, step 4.2: definition of a sequence of six segments that vary in temperature.

This assistance function is activated by default whenever the workflow of creating new wear tests is started. The reason for this is that the behavior of the forwarding function reflects the standard workflow at the author's laboratory. By turning it on by default, it greatly supports the laboratory policy on using sequential ids when individual test specimens are used in a multi-station tribometer and to use the same lot on each station when test specimen lots are used. This greatly reduces the number of opportunities at which a human can enter erroneously numbers even if

he is fundamentally aware of the correct number. Additionally, error correction by a human supervisor can be saved which is also highly error-prone process, see Table 4. As *Atlas*' JavaScript functionality and the templates for its HTML forms can be split across different files and even be re-created live from the database, other policies and workflows, e.g. of other laboratories can be easily implemented. After the first set of tests has been created, the form remains in the browser window by design. The user can then adjust the test specimen ids and re-submit the form. This process can be repeated the required number of times until all tests with the same specification have been created. This reduces the number of error-prone user operations to the possible minimum, i.e. every parameter has to be entered only once. Additionally, automatic functions exist that compute the suggested default values for selected test parameters. Specifically, when the user enters the sliding speed for a sliding segment, the recommended values for logging frequency and averaging time will be computed. This infusion of domain logic into automated functions obviates the user from making calls or calculations that have proven to exhibit especially high error-rates.

The number of user operations can be even further reduced when the user needs to create new tests that are replicates of existing tests. For this, *Atlas*' GUI provides a special form that allows the cloning of sets of existing tests. Again, by providing the tribometer in question, the id of a wear test and the id of a target project or test set, an asynchronous HTTP request is sent to the *Atlas* server which then generates a set of inputs for test specimen identifiers, one for each test position, pre-populate them with the ids of the original tests and send them back to the browser who will insert them into the page. The user can then inspect the test specimen ids of the source tests and either keep them, if he wants the new tests to use the same specimens than the source tests, or he can update them to make the new tests use other specimens.

4.8 Test setup and execution of wear tests

The process of conducting a wear test roughly consists of the steps shown in Figure 2. First, the test specimens need to be identified and mounted to the tribometer. *Atlas* supports this step by providing an overview of all tests that belong to a given project that includes information on all relevant test parameters, including which test specimens should be used for which test. Figure 24 shows a sample for such

a project-based overview. For clarity, tests are grouped by test set and segment index and sorted by test set id, segment index and wear test id.

Once identified, the test specimens are retrieved from storage which is greatly simplified when the storage system described in Section 4.5.4 is used, and then mounted to the corresponding tribometer position. Then any required auxiliary equipment, like lubricant pumps or heating and cooling devices are mounted. At the moment, the process of the selection of the required additional equipment depends on the appropriate knowledge of the laboratory personnel which in turn depends on proper staff training and process documentation (both of which having however non-negligible error rates, see Table 4).

Material pairing	Set Segment	Segment type	Loading	Applied weight	Sliding speed	Sliding distance	Duration [hh:mm:ss]	Set temp	Static COF rate	Static COF latency [hh:mm:ss]	Wear test	Test specimen	Wt	Wt loss	COF dyn,ss	ϕ_p [°C]	Status	Actions	
TECACOMP PEEK TRM XS black V0371-07/1 (B) vs. 100C6 (R)	378 / 1	Slide	5.000 MPa	20 N + 0 g	2.000 m/s	28800.0 m	04:00:00	50 °C	0.0	00:00:00	541968	27030 L164	-	-	-	-	D.U.E.	E.R.	
											541969	27031 L164	-	-	-	-	D.U.E.	E.R.	
											541970	27032 L164	-	-	-	-	D.U.E.	E.R.	
											541971	27033 L164	-	-	-	-	D.U.E.	E.R.	
											541972	27034 L164	-	-	-	-	D.U.E.	E.R.	
											541973	27035 L164	-	-	-	-	D.U.E.	E.R.	
											541974	27036 L164	-	-	-	-	D.U.E.	E.R.	
	541975	27037 L164	-	-	-	-	D.U.E.	E.R.											
	378 / 2	Slide	5.000 MPa	20 N + 0 g	2.000 m/s	28800.0 m	04:00:00	75 °C	0.0	00:00:00	541968	-	-	-	-	-	-	D.U.E.	E.R.
											541969	-	-	-	-	-	-	D.U.E.	E.R.
											541970	-	-	-	-	-	-	D.U.E.	E.R.
											541971	-	-	-	-	-	-	D.U.E.	E.R.
											541972	-	-	-	-	-	-	D.U.E.	E.R.
											541973	-	-	-	-	-	-	D.U.E.	E.R.
											541974	-	-	-	-	-	-	D.U.E.	E.R.
	541975	-	-	-	-	-	-	D.U.E.	E.R.										
378 / 3	Slide	5.000 MPa	20 N + 0 g	2.000 m/s	28800.0 m	04:00:00	100 °C	0.0	00:00:00	541968	-	-	-	-	-	-	D.U.E.	E.R.	
										541969	-	-	-	-	-	-	D.U.E.	E.R.	
										541970	-	-	-	-	-	-	D.U.E.	E.R.	
										541971	-	-	-	-	-	-	D.U.E.	E.R.	
										541972	-	-	-	-	-	-	D.U.E.	E.R.	
										541973	-	-	-	-	-	-	D.U.E.	E.R.	
										541974	-	-	-	-	-	-	D.U.E.	E.R.	
541975	-	-	-	-	-	-	D.U.E.	E.R.											
378 / 4	Slide	5.000 MPa	20 N + 0 g	2.000 m/s	21600.0 m	03:00:00	125 °C	0.0	00:00:00	541968	-	-	-	-	-	-	D.U.E.	E.R.	
										541969	-	-	-	-	-	-	D.U.E.	E.R.	
										541970	-	-	-	-	-	-	D.U.E.	E.R.	
										541971	-	-	-	-	-	-	D.U.E.	E.R.	
										541972	-	-	-	-	-	-	D.U.E.	E.R.	
										541973	-	-	-	-	-	-	D.U.E.	E.R.	
										541974	-	-	-	-	-	-	D.U.E.	E.R.	
541975	-	-	-	-	-	-	D.U.E.	E.R.											

Figure 24: HTML-view of tests that belong to a sample project, grouped by wear test sets and sorted by test set id, segment index and wear test id.

Once the test specimens are mounted and all required additional equipment has been identified and readied, the tribometer's control software needs to be parameterized correctly, i.e. in a way that it will execute the test as planned. Even tests with a low number of segments can more than dozens to more than one hundred parameters that need to be put into the tribometer control software. The state of the art on how this is supported by computerized tools greatly varies between tribometer manufacturers. Mostly, a large number of fields for the required test parameters is presented to the user with hardly any fill-in help or automatic validation

check or unit conversion (i.e. from rpm to v and vice versa or from absolute seconds to segment duration in hh:mm:ss format). Typically, a tribometer control software will offer the option to load an existing test, modify its parameters and save it as a new test. However, this process is very prone to a sequence of errors, amongst which omitting the necessary change of a parameter, making a required change but entering the wrong value, overwriting the chosen template or choosing the wrong test as a template should be the most frequent ones.

A proprietary solution that includes the assisted creating of wear tests and their automatic transmission to a tribometer is the TriboScript™ software of Bruker Corporation, which “utilizes a drag-and-drop methodology to prepare test scripts” and which is able to transmit these offline-prepared scripts to a Bruker tribometer for execution [19]. Currently, no published literature contains estimates on error rates of elementary processes in a tribology laboratory in general or for the process of tribometer parameterization. Without conducting any structured process hazard analysis, but based on his years long experience as head of a tribology laboratory, the author’s personal estimate for the probability of making at least one non-recoverable error when configuring a tribometer control software, i.e. making an error that mandates the repetition of the test, is 5-10 % for single-segment tests with uniform sliding and 10-20 % for multi-segment tests that include oscillation, static COF or cyclic positioning segments. As these error rates are unacceptably high, *Atlas* provides a functionality for the automatic transmission of test parameters to a tribometer for which Figure 26 shows the basic workflow. Basically, *Atlas* supports three modes of automatic parameterization: online pull, online push and offline. In pull mode, the ids of the test(s) are entered into the tribometer control software which then sends a corresponding HTTP request to the *Atlas* server. For this to work, the tribometer control software needs to support this functionality. It needs to provide inputs for the wear test id(s), needs to issue the HTTP request, thereby supplying all information needed by *Atlas* and finally, it needs to be able to understand and process the returned configuration which is send as part of the HTTP response. The GET HTTP request should be made to `/wear_tests/autoconfig.[format]`, where [format] is the expected response format. Currently only “txt” for a plain text response is available which is the expected document format of the *Atlas TT* tribometers of Tribologic GmbH (Germany) and of IVW – which are the only tribometers that are currently designed for use with *Atlas*’ auto-configuration function.

When the request completes successful, i.e. with a status code in the 200-299 range, the server's response will include a document of the requested format in its body. The tribometer control software is then expected to parse this document and to extract all required information to execute the test. The tribometer software needs to include the following mandatory information as Common Gateway Interface (CGI) parameters: the `tribometer_id` of the corresponding tribometer record on the *Atlas* server that the request is sent to and one set of key value pairs for each combination of position (key) and wear test (value). The key must be equal to the value of the `pos_no` attribute of the corresponding position's *Atlas* record and the value must be equal to the id of the *Atlas* record of the wear test that is to be done on the correspond-

ing position. The relative URL for requesting a text-formatted configuration document for the execution of wear test 786 on position 1 and wear test 787 on position 2 of the tribometer with id 3, would be `/wear_tests/autoconfig.txt?tribometer_id=3&1=786&2=787`.

The workflow for push and offline mode both start with entering the above-mentioned information into the form shown in Figure 26, by selecting the required mode (download or push) and by clicking on "OK". *Atlas* will then generate the requested document and either send it as a download to the user's browser (offline mode) or directly to the tribometer's control PC via an HTTP POST-request (push). In the latter case, the tribometer control software needs to return a corresponding response that at least indicates the result of the parsing process, e.g. a 201 status

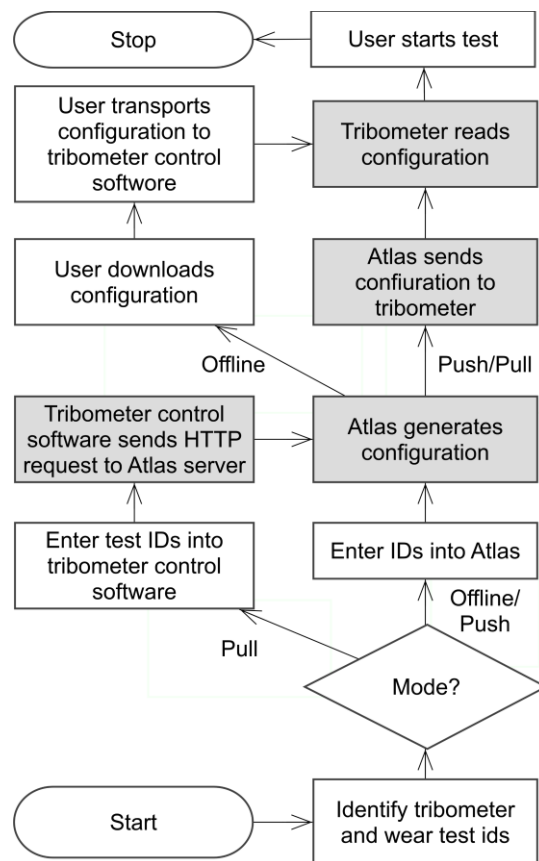


Figure 25: Automatic configuration of a tribometer with *Atlas*. Grey: computerized support. Method Support is subject to support from the tribometer control software.

code if the test has been set-up correctly. When the user has chosen to download the document, he needs to transport it manually to the tribometer control software. The document is then injected into the tribometer control software using whatever process is provided by it. Finally, when the configuration has been parsed by the tribometer control software and if the test has been set up correctly, the user starts the test.

4.9 Measurement data files

4.9.1 Domain

The main source of measurement data files are tribometers which produce them over the course of an experiment. Given the huge range of brands and models, they come at an equal range of data formats and with different naming strategies. As for naming strategies, the data file name is typically a string whose length and allowed characters are determined by the underlying operating system. Therefore, the user is typically provided with a file select dialogue that is provided by the operating system. Tribometer operators then need to define a set of rules for creating “meaningful names” for data files. Such rules then specify what information should be included in the file name so that the file’s content can be safely related to a specific test. In other words, the purpose of the file name is to serve as an identifier for the file itself and to associate it with the corresponding experiment. Mostly, this is done without correctly defining the domain of the task and its entities and relationships explicitly, without defining a data model for this and without defining identifiers for entities that should be associated with the data file, i.e. operators, test devices, test specimens, project name or id etc. Often, the inclusion of metadata like recording date, file version or file format are also specified. Additionally, the use of certain characters can be mandated or excluded, like special characters or characters that serve special purposes on certain operating systems (e.g. on Linux, a leading “.” Indicates a “hidden” file or directory) or that are customarily reserved

Push wear test(s) to tribomete

Autocomplete test IDs

Tribometer

Test ids

Position 1

Wear Test ID

Mode

Push to tribometer

Figure 26: Form for starting a push workflow for tribometer configuration.

to indicate special roles (e.g. trailing tilde for automatic backup files). Additional rules then cover the location at which a file should be stored. Typical information that needs to be minded when deriving storage location are project, sample name, backup priority, investigator/operator and rules concerning internal folder structure of storage locations. All those rules together form a “policy on data file naming and storage locations”. Even being only a part of an even wider set of rules (“Research Data Management Policy”), they can easily fill dozens of pages. This policy needs to be published to the laboratory personnel, be reviewed and updated periodically and the personnel needs to train its application, including processes for arbitrating ambiguities and non-included edge cases. Although a huge step up from completely unregulated naming and storing of data files, or more general “assets”, this process is still prone to significant error rates, see Table 4. Therefore, laboratories should consider whether the improvements achieved as a result of the introduction and operation of a file name and storage location policy actually exceed the costs. Measurement data files also come in different states of processing. The two basic processing state are “as generated by a tribometer” and “evaluated”. In-between, there might be none, one or more steps of partial processing or format conversion. Such intermediary states may be either completely transient in-memory states or they might be persisted to a file storage system.

4.9.2 Uniform data format and naming convention

Today, tribometers are computer-controlled and equipped with an array of sensors for a set of physical quantities, e.g. friction force, normal force, test specimen temperature or ambient humidity. On multi test-station tribometers, some of these may be recorded for each test position individually and some may be recorded as global variables that apply to each test position in the same way. The classification of a specific physical quantity of being either test position-specific or global may depend on tribometer design. For the two construction variants of four position-tribometers shown in Figure 12 ambient humidity is a global parameter for the block on ring-variant because the four positions are operated in the laboratory atmosphere, whose humidity and temperature therefore apply to each position equally. For the pin on disc-variant, which exhibits individual climate chambers for each test position, ambient humidity and temperature can potentially be different for each test position.

The simplest approach that covers all potential combinations of physical quantities and their classification as either position-specific or global is to produce one write all recorded data into a single data file that contains recorded data in columnar format, with one recorded quantity per column. For position-specific quantities this results in a data file that contains multiple columns for the same physical quantity, e.g. friction force, with one column per test position. This – very popular – data file format is however at odds with the concept of individual tests by producing a joint data file for otherwise independent tribological tests. Specifically, it generates the risk of race conditions in which a process that executes an operation on one of the tests whose data is included in one given data file, e.g. computing an average temperature, may collide with another process that operates on a different test that, by chance or not, has its data stored in the same data file. Such race conditions can cause unpredictable results including data corruption, data loss, unhandled exceptions that cause the system to crash and, even more dangerous, unnoticed obfuscation of data in-between tests. While preventing such race conditions is basically possible, it is much easier to design the data file format in a way that prevents such race conditions by design. One simple way to do so, is to write a separate data file for each test that is done in parallel on a multi-position tribometer. This makes it necessary either duplicate global data by writing it to each separate file or to write a separate data file that only includes global test data.

Nummer	Zeit s	Reibungskraft		Belastung mm	Reibungszahl		Verschleiss		Temperatur V	
		N	N		°C	mm	V	V		
0	0,000	0,12	5,0	0,02370	-0,0001	28,7	0,041	3,709	0,123	2,513
1	0,100	0,07	5,0	0,01421	-0,0032	28,7	0,041	3,709	0,123	2,513
2	0,200	-0,01	5,0	-0,00222	-0,0080	28,7	0,041	3,709	0,123	2,513
3	0,300	-0,01	5,0	-0,00295	-0,0079	28,7	0,041	3,709	0,123	2,513
4	0,400	0,01	5,0	0,00288	-0,0081	28,7	0,041	3,709	0,123	2,513
5	0,500	0,01	5,0	0,00178	-0,0080	28,7	0,041	3,709	0,123	2,513
6	0,600	0,00	5,0	-0,00031	-0,0078	28,7	0,041	3,709	0,123	2,513
7	0,700	-0,16	5,0	-0,03170	-0,0068	28,7	0,044	3,709	0,123	2,513

Figure 27: Example of a tab-separated columnar data file from a tribometer. The column width exceeds the viewer tab width at several occasions, resulting in partial shift of columns.

While the first approach is “just” verbose and requires more storage space, the latter approach is – again – prone to race conditions and to increased overhead for establishing and maintaining the relationship between the potential joint data file

and the collection of tests that have their global data stored in this file. Out of these choices, *Atlas* creates an individual data file for each test that is done in parallel on a multi-position tribometer and it duplicates global data in each of these files. Thereby it trades minimum data file size for adherence to the encapsulation principle which in turn results in an inherently increased robustness to race conditions.

Once test data is recorded and stored in data files, provisioning of automated functions for handling, analyzing and plotting this data requires the LIMS to know the exact storage location of each physical quantity. This necessitates that the LIMS knows the name and the storage location of the data file that corresponds to a given test (together, they form the storage path of the test's data) and the exact storage location of the physical quantity in question within that file. For columnar data, this means knowing the corresponding column index. The most common solution to this is to use one or more "header lines" which contain human readable identifiers for the physical quantities that are stored in the corresponding columns and for their physical unit. When the unit is included, it may either be on the same line than the quantity or in a separate column, see Figure 27:

In order to be usable in an automated way, e.g. for automated data handling, processing, aggregation or plotting, a machine-readable relationship must be established between the file's identifiers (and unit labels) and the corresponding business logic entities ("friction force" and "loading"). While this is easy to do for an individual file or for a type of homogeneous files that were produced by the same tribometer, it becomes a more complicated task when data files exhibit incompati-

```

---
:wear_test_id: 546285
:column_descriptor:
  :date:
    :index: 0
    :name: Date
    :unit: ''
    :null: 0.0
    :precision: 0
  :time:
    :index: 1
    :name: Time
    :unit: ''
    :null: 0.0
    :precision: 0
  :sliding_speed:
    :index: 2
    :name: Sliding Speed
    :unit: m/s
    :null: 0.0
    :precision: 2
  :ambient_temperature:
    :index: 3
    :name: Ambient Temperature
    :unit: degC
    :null: 0.0
    :precision: 1
  :ambient_humidity:
    :index: 4
    :name: Ambient Humidity
    :unit: percent
    :null: 0.0
    :precision: 1
  :counter_body_temperature:
    :index: 5
    :name: Counter Body Temperature
    :unit: degC
    :null: 0.0
    :precision: 1

```

Figure 28: Example of a YAML-file that contains an *Atlas* data file column descriptor.

ble versions (for example when the tribometer's set of sensors is altered or its control software is change) or if they originate from different tribometers, potentially even from different brands and manufacturers. The advantage of such a key-value storage is that it can be easily expanded by new key-value pairs. For example, storing the minimum and maximum values of each column can be used for quickly determining plotting range without having to re-parse the actual data. Other interesting metadata include the presence or absence of a column, i.e. due to a missing or disabled sensor, or the validity of the data, e.g. when a sensor has ex-post been found to have been recording wrong data. In order to enable functionalities that require the stored metadata without having to re-read the whole data file from disk, *Atlas* stores its data file descriptors in separate, small files on disk (with the database itself as an even faster alternative). For doing so, a variety of data formats is available, e.g. the Extensible Markup Language (XML) [56], JavaScript Object Notation (JSON) Data Interchange Format [57] and the "YAML Ain't Markup Language" (YAML) [58] of which YAML was chosen for *Atlas* due to its support by Ruby's core library [59].

In order to work properly, in addition to the information from the data file header, transformation scripts will typically require meta-information about the test, test specimen, testing conditions and/or tribometer. From within *Atlas*, all this information is easily available by using the entity relationship diagram to retrieve the required objects – either by directly looking them up if their ID is known, or by following the specified relationships to retrieve them as associated objects. Once the correct object is found, it can be queried for the corresponding attribute. While, via the API, this is also possible from outside *Atlas*, it is much easier to do from within. "From within" in this case means in-memory objects of an *Atlas* server instance. However, as different tribometers produce different raw, i.e. non-normalized, data files, highly specific transformation scripts are needed. Here, specificity at least includes tribometer brand or even ID and possibly even data file version (unless it can be included in a single multi-version script).

Overall, specific scripts require specific information from the LIMS in order to convert a tribometer-generated data file into an internal data file that complies with a predefined, uniform format. *Atlas* solves this issue by providing an individual `pre-process` method to each instance of the tribometer class using the OOP concept

of “singleton methods”, which are instance-specific methods or method implementations. Code listing 18 shows the definition of singleton methods for two tribometers instances.

Code listing 18: Definition of singleton-methods for two tribometer instances.

```
# create two new tribometer and print their IDs
tribometer_1 = Tribometer.create({ name: "XTM 1000" })
tribometer_1.id
=> 1
tribometer_2 = Tribometer.create({ name: "SRV 5" })
tribometer_2.id
=> 2
# define singleton methods that will be different for both instances
tribometer_1.define_singleton_method(:say_hello) do
  puts "Hello, this is tribometer #{self.id}!"
end
tribometer_2.define_singleton_method(:say_hello) do
  puts "Bonjour, c'est le tribomètre #{self.id} ici!"
end
# call method for both tribometer instances
tribometer_1.say_hello
=> Hello, this is tribometer 1!
Tribometer_2.say_hello
=> Bonjour, c'est le tribomètre 2 ici!
```

The advantage of this individualization is that both instances have full access to *Atlas'* data, as can be seen from the inclusion of the instance IDs in the output of Code listing 18. Specifically, this access is also provided to the code block that is passed as an argument to the call of the `define_singleton_method` method (which all Rails instance objects inherit from Ruby's `Object` class by default).

Code listing 19: Dynamic definition of a singleton-method that transforms a data file for `wear_test` located at `file_path` to standard format.

```
Class Tribometer
  after_initialize do |tribometer|
    # define singleton method for this instance
    tribometer.define_singleton_method(:preprocess) do
      eval tribometer.preprocess_code
    end
  end
end
```

Next, each tribometer must be given its own singleton method which will host the specific domain logic of the respective tribometer and the data files produced by it. In order to prevent the inclusion of their implementation into *Atlas'* source code

(which would lead to collisions between different server instances), the source code of the transformation scripts is persisted in an attribute of the tribometer model called `preprocess_code`, whose content is then used to dynamically create an internal preprocess singleton method which is then executed, see Code listing 19.

By convention, each preprocessing code is expected to return a transformed data file that complies to *Atlas*' standard file format. This requires the encoding of the file to be UTF-8 and the content to be tab-separated columns of data, numeric values to be either in scientific, exponential or engineering format (with '.' as decimal separator) and "\n" as newline character. Furthermore, one file per segment is required, with the file name being "`\d{8}.\d{4}.\s+`", where `\d{8}` is the eight-digit wear test id (filled-up to eight digits with leading zeroes), `\d{4}` is the four-digit segment index (again, filled-up to four digits with leading zeroes) and `\s+` representing the files processing state. By default, *Atlas* supports three different processing state identifiers:

- "dat" for data files which have either been produced by tribometers that natively support *Atlas*' standard data file format or which have been converted from non-complying file formats during upload (see Section 4.9.6).
- "orig" for data files that have experienced partial preprocessing but which are not yet fully evaluated. Possible reasons for this is the conversion of timestamps to elapsed test time or sliding distance data (see Section 4.9.6), or the insertion of external data into a "dat" file for forming consolidated data for further processing (see Section 5.4).
- "proc" for fully processed and evaluated data files. Typically, files in this state contain data that is used for generation plots of recorded and computed quantities as a function of elapsed test time and for the computation of aggregated data, e.g. of mean coefficients of friction or of wear rates in steady-state, see Section 4.9.7)

Additionally, for each processing state, *Atlas* expects accompanying (but separate) files that contain the machine-readable file column descriptor, see Figure 28.

4.9.3 Type specific domain requirements

Different types of wear test segments furthermore require data files with different special characteristics. A typical example for this is oscillatory sliding motion which is modeled by the `OscillationSegment` class. Oscillatory motion is characterized by the existence of two reversal points. In close spatial and temporal vicinity to these turning points, the resolution and accuracy of the data acquisition is often not good enough to always yield valid data, especially concerning the friction force. Specifically, any combination of sensor sampling frequency and logging interval will lead to the situation in which there is one logging interval that will partially include forwards motion, the reversal point and backwards motion. With the typical convention of recording friction forces as positive in one direction and as negative in the other direction, the resulting average friction force that is recorded for this logging interval will be lower than the friction force that was in effect actually. Still, most of times, data gets written to the measurement data file in one long sequence of rows, with one row representing each logging interval and without any logical markers for the reversal points. This loss of procedural information imposes great difficulties to the evaluation procedure when it has to identify the data points that need to be excluded from certain operations, e.g. when computing an arithmetic mean for the coefficient of friction or when plotting the coefficient of friction as a function of test elapsed time or sliding distance.

Therefore, *Atlas* supports features that reflect type-specific features of measurement datafiles. An example for this is the inclusion of reversal point markers (“### period x”) in data files that represent oscillation segment, where x is the number of the oscillation cycle of the data below, see Figure 29.

```
### pass 0001
### period 1.0
14.11.2018 15:10:08:393 0.00 17.9 48.0 21.1 0.002 0.011 -0.000
14.11.2018 15:10:09:393 0.02 17.9 48.0 21.0 -0.004 0.026 0.016
14.11.2018 15:10:10:393 0.99 17.8 48.0 21.0 0.007 0.025 0.159
14.11.2018 15:10:11:393 1.00 17.8 48.0 21.1 -0.003 0.044 0.302
14.11.2018 15:10:12:394 0.99 17.9 48.1 21.2 0.013 0.016 0.445
### period 1.5
14.11.2018 15:10:13:393 1.00 17.9 48.1 21.2 -0.002 0.022 0.588
14.11.2018 15:10:14:393 0.99 17.9 48.1 21.2 -0.000 -0.201 0.731
14.11.2018 15:10:15:393 0.99 17.8 48.1 21.1 -0.018 -0.743 0.874
14.11.2018 15:10:16:401 0.99 17.9 48.1 21.1 -0.011 -0.606 1.017
14.11.2018 15:10:17:393 0.99 17.9 48.1 21.1 0.010 -1.278 1.159
### period 2.0
14.11.2018 15:10:18:394 0.99 17.8 48.1 21.2 0.004 -0.589 1.302
14.11.2018 15:10:19:393 1.00 17.9 48.1 21.2 -0.002 -0.714 1.324
```

Figure 29: Use of reversal point markers for segmenting data files that represent oscillation segments.

For tribometers that do not implement such reversal markers, *Atlas* provides functionalities that will identify reversal points in continuous streams of data. This feature needs to be implemented as part of the tribometer-specific conversion scripts, see Code listing 18.

Other type-specific features that mandate or support the use logical markers for segmenting a wear test segment's data into meaningful subsegments are the positions that form the skeleton of the elementary motion pattern that is cyclically repeated as part of a `CyclicPositioningSegment`.

4.9.4 Data model

Atlas implements a `Datafile` class to implement the general domain logic of data files (i.e. save to disc, read from disc, standardize data file name, on-server storage location etc.). The domain logic of measurement data files that belong to individual wear test segments are then generically modelled by `SegmentRecordedDatafile` which is a subclass of `Datafile`. Then, on the next level of hierarchy, a group of sibling subclasses is defined which represent the different types of wear tests segments, see Table 3, and which can be used to override existing methods of the `SegmentRecordedDatafile` class, i.e. to reflect specifics of the file's domain logic like an internal data structure that is different to that used by other segment types, or to add new methods that implement domain logic that is exclusive to the respective segment type and its measurement data files.

`Datafile` itself all its ancestors are no subclasses of `ActiveRecord` and their instances are therefore not persisted to the database. Instead, they are generated on-the-fly by a custom attribute reader and returned as its return value. This establishes a relationship between the class that implements this reader and the datafile class whose instances are generated. Code listing 20 shows this using the example of the relationship between `WtSegment` and `SegmentRecordedDataFile`. When a test segment requests its datafile object, the exact class of the returned object is determined from the subclass of the segment ("`self.type`") and the requested processing state ("`type`").

Code listing 20: Association between wear tests and data files.

```

Class WtSegment < ApplicationRecord
  def datafile_class(type)
    "#{self.type}#{type.to_s.capitalize}Datafile".constantize
  end
  def datafile(type)
    type = type.to_sym
    @datafiles ||= {}.with_indifferent_access
    if @datafiles[type].blank?
      @datafiles[type] = datafile_class(type).new({ wt_segment: self })
    end
    @datafiles[type]
  end
end

class SegmentRecordedDatafile < Datafile
  attr_accessor :wt_segment
  def initialize(wt_segment: nil # more parameters omitted for clarity)
    @wt_segment = wt_segment
    # more code omitted
  end
end

```

The services that datafile objects provide include the loading and saving of data from and to disk, storing data while it is in-memory, accessing columnar data based on standardized quantity identifiers, see Code listing 21.

Code listing 21: Using datafile instances to load data from disk and to get columnar data for measured ambient humidity.

```

Slide_segment = SlideSegment.last
slide_segment.datafile(:proc).load
proc_df.get_data_for :ambient_humidity
=> # array holding friction force data
proc_df.get_unit_for :ambient_humidity
=> "percent"

```

4.9.5 Server-side storage

Atlas uses a model of its own for storing assets on the server-side called `AssetFolder`. This model encapsulates all on-disk processes like, writing of a new file, transferring uploaded files to their storage location, finding out whether a requested file exists and if so, reading it. When associated to another model, they represent the on-disk storage locations for all data that is associated with this model but that should not be persisted in the database. Like instances of `Datafile`, instances of

`AssetFolder` do not inherit from `ActiveRecord::Base`. Besides not being persisted to the database, they cannot be associated to other objects using Rails built-in domain specific language for declaring relationships. Instead, like with datafiles, their association to other objects is established by creating a corresponding attribute reader that constructs the requested instance on-the-fly, see Code listing 22.

Code listing 22: Association between wear tests and asset folders.

```

class ApplicationRecord < ActiveRecord::Base
  after_create :create_asset_folder
  after_destroy :destroy_asset_folder
  # create virtual association
  def asset_folder
    @af ||= AssetFolder.new(owner: self)
  end
  def create_asset_folder
    asset_folder.create
  end
end

def asset_folder=(af)
  @af = af
end

def destroy_asset_folder
  asset_folder.destroy
end

# inherit asset folder to subclasses of ApplicationRecord
class WearTest < ApplicationRecord
end

WearTest.sample.asset_folder.relative_path
=> "/wear_tests/00001000-00001999/00001724/"

```

The most important functionality of `AssetFolder` is the derivation of the storage location for all data that is handed to it. This enables *Atlas* to consistently read and save assets of any given object to an identifiable and consistent location on the server's storage. Code listing 23 demonstrates this using the example of composing the on-disk storage path for an instance of the `Attachment` class.

Code listing 23: `AssetFolder`

```

class Attachment
  def file_path
    File.join asset_folder.path,
              file_name_on_hdd
  end
end

```

The root of *Atlas* physical storage is part of the configuration of the respective server instance and may be either a machine-local storage or a remote storage that is integrated into the server's file system, e.g. a network file system resource.

4.9.6 Data file upload and automatic standardization

Before a measurement data file can be stored on the server, it must be uploaded to it, see Figure 30. As *Atlas* does not provide direct access to its data storage system, data files can only be pushed to the server by an HTTP request. The relative URL is `/wear_tests/[ID]/upload_datafile.html`, with [ID] being the id of the wear test for which one or more data file(s) should be uploaded. As described in Section 4.9.1, *Atlas* stores measured test data in one separate file per test segment and the data file header in another separate file. The exact number of files that needs to be uploaded therefore depends on the number of a test's segments. In order to support this, *Atlas* will accept a zip file as part of the POST request which it will unpack and check for its content. When already in standard format, the unpacked files will be sent to their asset folder. Data files which are not in standard format, will be converted on the fly to *Atlas*' standard data file format. Then, the converted data file(s) are moved to storage. *Atlas* does not persist any uploaded data files that are not in standard format and discards them. Log files that comply to *Atlas* naming convention for this file type will also be transferred to storage. See Section 4.9.5 for a detailed description on how *Atlas* determines a data files' on-server storage location.

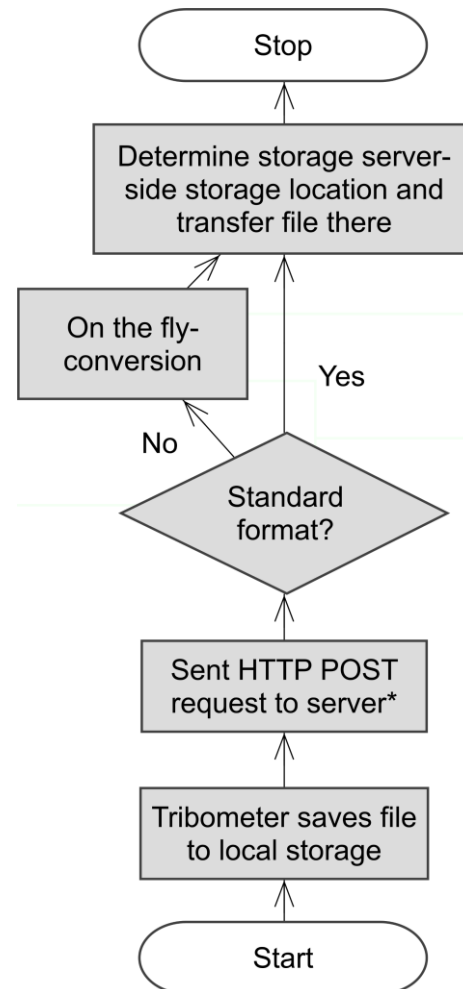


Figure 30: Workflow for uploading measurement data files. Grey: computerized support. *Level of support depends on machine.

4.9.7 Automatic evaluation and result persistence

One of the major challenges in evaluating measurement data files programmatically, i.e. by the use of software-coded procedures, is the high variability of input data. This already starts with such fundamental data as the “independent variable”

which is most frequently “time”. Possible data formats include decimal notation in all kinds of units (mostly seconds or hours) or the HH:MM:SS format. When representing “test elapsed time”, this kind of data starts at “zero” at the beginning of the test, i.e. at 0 or at 00:00:00. Furthermore, time data can be expressed as timestamp, in which case they refer to an absolute time scale. They come in an even wider variety of data formats, for example as a numeric value that represents the elapsed time since the beginning of an “epoch” (e.g. since January 01, 1970 UTC in the case of Unix time stamps) or in an alphanumeric format that explicitly defines a point in time in a given time zone (e.g. “2021-12-13T14:22:26+00:00” according to ISO 8601). The number of possible permutations, and thereby complexity, is further compounded by the virtually infinite numbers of physical quantities that can be included in a data file, the column they are stored in and the range of physical units they can be stored in.

Atlas solves this issue by converting uploaded measurement data files to a uniform internal data format, combined with a machine-readable data file header. The required domain knowledge that is needed to achieve this uniformity is hereby included in the tribometer-specific conversion scripts that are stored in the `preprocess_code` of the tribometer instances.

The second major challenge is the high variability of evaluation procedures and of the results that they produce. While some types of experiments enable the calculation of test specimen wear from sensor data, e.g. the calculation of the linear wear rate from time-resolved data on the height loss of test specimens, others do not allow this specific calculation. There are similar situations for other result quantities, test types and even tribometers, e.g. when one tribometer tracks test specimen temperatures while another one does not. Additional variability arises from the different types of test segments that a wear test can be made of. While in uniform sliding the average dynamic coefficient of friction in steady state typically is a relevant result, this quantity is not even defined for a static COF segment. *Atlas* mainly uses the OOP concept of specialization to mirror this variability: Whenever a procedure, e.g. the calculation of a contact area as a function of height loss, Depends on the test geometry, this procedure is implemented in the subtypes of the `Wear-Test` class. Each subclass can then use its specific domain knowledge to implement the appropriate procedure.

The same strategy is used for implementing differences that arise from different types of wear test segments, see Code listing 24. In this, all the shown segment types implement different calculations of the sliding speed using the conversion library for physical quantities of Olbrich [60]. For a sliding segment instance this is a straight-forward construction of a unit string from its attributes `sliding_speed` and `sliding_speed_unit` and its subsequent interpretation as a `Unit` object [60] and conversion to the unit specified by the corresponding argument.

Code listing 24: Implementation of type specific calculations of the same information using specialization.

```

class OscillateSegment < WtSegment
  def sliding_speed(unit: "m/s")
    f = U "#{oscillation_frequency} #{oscillation_frequency_unit}"
    s = U "#{stroke} #{stroke_unit}"
    v = f * s * 2.0
    v.convert_to(unit).scalar.to_f
  end
end

class SlidingSegment < WtSegment
  def sliding_speed(unit: "m/s")
    v = U "#{sliding_speed} #{sliding_speed_unit}"
    v.convert_to(unit).scalar.to_f
  end
end

class StopSegment < WtSegment
  def sliding_speed(unit: "m/s")
    return 0.0
  end
end

s = SlidingSegment.new({sliding_speed: 5, sliding_speed_unit: "mm/s"})
s.sliding_speed(unit: "cm/min") => 30.0

```

For an oscillation segment instance, the construction of two physical quantities as `Unit` instances is needed, which are then multiplied, converted to the desired output unit and then the absolute value is calculated and returned as a float. For stop segment instances, the implementation is trivial. In all three implementations, the methods exhibit the same name and the same set of parameters and return values of the same type. Therefore, all three are called identically, can be supplied the same values as parameters and will yield the same return value type. In addition to the shown implementations, *Atlas* also has implementations for instances of the `CyclicPositionSegment` and `OscillateSegment` classes. As a consequence, any code that will query an instance of any segment type does not need

to check whether it will respond to a call to `sliding_speed` and what type of value it might return because this method is implemented for all existing subtypes of the `WtSegment` class. Furthermore, by defaulting the return value to be in m/s unless explicitly stated otherwise, all code occurrences that do not specify another unit can be sure to get a value in m/s.

Using these techniques to reduce domain complexity, *Atlas* is able to automatically evaluate test data with a small set of pre-determined identifiers for physical quantities and evaluation procedures. By default, evaluation is conducted by calling the instance method `evaluate` of a wear test or a wear test segment. Then, within the `evaluate` method this segment, its domain knowledge can be translated into code. If a segment type typically computes an arithmetic mean of the dynamic coefficient of friction, the respective calculation can be conducted and the result can be stored in the respective attribute of the segment instance. In order to enforce the “don’t repeat yourself” paradigm, such “atomic” building block of an evaluation are written as isolated procedures that use an interface that exhibits the generalization that is required for the procedure to be used by different types of callers and by passing-in parameters with such a degree of standardization, that the internals of the procedure do not need to worry about type-specifics (which should be handled in type-specific code sections).

Code listing 25 shows the implementation of such a method. Its first argument is the identifier of the quantity that should be computed. By design, the quantity is derived from the name of the attribute in which the calculation result should be persisted. A second attribute is a data file object, i.e. an instance of any of *Atlas*’ data file classes. This attribute is optional, as not all quantities might require access to the content of a data file (e.g. the calculation of mass loss which only requires the initial and final mass of the respective test specimen, both of which are persisted as attributes of segment instances). The specific calculations for the various attributes are then included in a branch of a case switch (other potential implementations include code blocks as the value of a key-value storage with the keys being the attribute names). When the calculation is completed (and quality checked, e.g. for infinite numbers), the result is persisted to the specified attribute and returned to the caller.

Code listing 25: Implementation of a method for performing elementary calculations during the evaluation of wear test segments. For clarity, only one such calculation is shown. Other calculations can be included by adding new branches to the case statement.

```

Class WtSegment < ApplicationRecord
  def compute(attr, df=nil)
    val = case attr.to_sym
      when :cof_mean
        df = datafile(:proc) if df.blank?
        if df.has_data_for? :cof
          data_A = df.get_data_for :cof
          get_eval_interval_data(:cof_avg,data_A).arithmetic_mean
        else
          nil
        end
      end
    val = nil if val.numeric? && (not val.to_f.finite?)
    self.send "#{attr}=", val
  end
end

```

In order to prevent code duplication, this instance method is defined for the `WtSegment` class, which is the superclass of all type-specific segment classes in *Atlas*, and then inherited to its type-specific subclasses. When these want to override only parts of this implementation, they can do so via Ruby's `super` keyword, see Code listing 26.

Code listing 26: Partial overriding of the `compute` method to reflect segment-type specific domain logic without overriding all functionality of the superclass' implementation.

```

Class StaticCofSegment < WtSegment
  def compute(attr, df=nil)
    case attr.to_sym
      when :cof_mean
        self.cof_mean = 0
        # other type specific branches
      else
        # delegate calls to all other attributes to the overridden method
        super
      end
    end
  end
end

```

Having used inheritance, specialization, overriding and back-referral to overridden methods to define type-specifics, these building blocks can be used to compose

segment type-specific definitions of the evaluation method. When a calculation produces not (only) an aggregated result that should be stored in a single-value attribute but (also) a new data column that should be stored in the datafile that represents evaluated data, it must be stored in the corresponding data file object. After all columnar data has been added, the data should be written to disk.

Code listing 27: Calculation of columnar data for the coefficient of friction, its storage in a “proc”-type data file object and persistence of the data file object to disk.

```
# load pre-processed data from disk
orig_df = datafile(:orig, autoload: true)
# create new data file object for processed data
proc_df = datafile(:proc)
# get friction force data column
friction_force_data = orig_df.get_data_for :corrected_friction_force,
                                          unit: 'N'

# get normal force
normal_force = self.loading(unit: 'N')
cof_data = friction_force_data.divide_by normal_force
# create entry for data file column descriptor of proc data file
cd = { name: 'Coefficient of friction',
      unit: '1'
    }
proc_df.set_data_for :cof, data: cof_data, cd: cd
# make other calculations, add more columns and then save data to disk
proc_df.save
```

4.10 Visualization of results

Once wear test data is evaluated and stored, the results need be visualized, e.g. for inspection, quality control and approval by the user or for being used in a project report or in a scientific publication. Important ways of visualizing result data are tables and graphs. For generating tables, *Atlas* renders HTML table elements.

Depending on the type of result and level of aggregation, *Atlas* provides different visualization tools. Aggregation levels include individual wear test, wear test set and project. On the level of individual results, *Atlas* provides a detailed overview of all data that is associated with an individual test in HTML format, including test parameters, metadata, numeric test results, user comments, results of profilometric measurements or any other data in the form of attachments (e.g. optical micrographs, photographs of the test setup, SEM-images etc).

On this page, controls for creating graphs of columnar data are included. For each wear test segment, the user can create an arbitrary number of graphs. For this he can either generate a generic graph for which no settings are predefined, or, as an assistance, predefined graphs (so called “prototypes”) can be created for which the type of quantity that is shown on each of the four plot axes (x, x2, y and y2) is predefined. When any of these

options is chosen, an associated JavaScript function will send a corresponding asynchronous request to the *Atlas* server, who will create the requested graph object, render its HTML representation and send this back to the browser, who will then insert the graph into the existing page. The user can then modify the default settings (if any) and replot the graph, see Figure 32. The set of quantities that can be plotted

Add recorded data graph

Recorded data graph prototypes

ΔV (y1) and COF (y2) vs. sliding distance

Δh (y1) and F_f (y2) vs. sliding distance

ϑ_{cb} (y1) and ϑ_{gear} (y2) vs. sliding distance

rH (y1) and ϑ_{amb} (y2) vs. sliding distance

Static COF plot (= COF (y1) and s (y2) vs. t)

Luminosity analysis prototypes:

in-track (y1) and ref. (y2) luminosity vs. sliding distance

COF (y1) and ΔL_{rel} (y2) vs. sliding distance

Δh (y1) and ϑ_{cb} (y2) vs. sliding distance

Multi y-axis plots (experimental, use with caution)

ΔV (y1), COF (y2) and ϑ_{cb} (y3) vs. s

Generic recorded data graph

Generic

Figure 31: Controls for creating a new graph for recorded data (and columnar data derived from it).

is dynamically retrieved from the column descriptor of the “proc”-type data file of the corresponding segment. This dynamic population keeps the set of option up to date if new quantities are added to the data file (see Section 5.4 for an example) and makes sure that only quantities can be selected for which data are actually available.

Even with a small number of tests, and even with the shortcuts that are provided for predefined graph types, the task of creating result plots can become very repetitive (and therefore error-prone) very quickly. Most of times, the selection of results plot for individual tests is the same within a project, or at least within a larger testing campaign. For this, *Atlas* provides the option to pre-select all, any or none of the predefined graphs directly when creating tests.

115674

[clone](#) [download](#) [destroy](#) [replot](#) [reset](#) [settings](#)



General settings										
Title	- no title -									
Width x Height	600 x 400	Use default colors?	Yes	Frozen?	No					
Style	Lines only	Use soft 0 for FF/COF?	Yes							
Font scale	100 %	Plot key?	No							
Imprint id?	auto	Key location (h/v)	default							
Type specific settings										
Plot volume loss regression curve?	Y	Plot volume loss regression range?	Y							
Plot admissible ambient temp./hum.?	N	Plot CB temp. eval range?	Y							
Plot COF eval range?	Y	Plot TF lum eval range?	Y							
Smooth	- none -									
Axis settings										
Axis	Use?	Physical quantity	Unit	Label	Range	In steps of	Range divide by	Log10-scale?	Precision	Color
X	Yes	segment_sliding_distance	km	label	from...to	steps	1.0	linear	auto	n.a.
Y	Yes	volume_loss	as data	label	from...to	steps	n.a.	linear	auto	
Y2	Auto	cof	as data	label	from...to	steps	n.a.	linear	auto	
Z	Auto	- none -	as data	label	from...to	steps	n.a.	linear	auto	n.a.
X2	Auto	segment_elapsed_time	as data	label	from...to	steps	n.a.	linear	auto	n.a.
Info										
Version no: 1										

Figure 32: Form for changing the settings of a recorded data graph.

As with data files, *Atlas* uses a hierarchical data model for graphs, see Figure 39, in order to reflect the variability of graph types and the variability that this induces on the specifics of attributes and behavior. The two major types of graphs in *Atlas* are recorded data graphs and graphs of aggregated data. Here, each subclass fulfills its own specific purpose. Graphs that plot columnar sensor data or data computed from it are modelled by instances of the `SegmentRecordedDatafile` class. There are three further specializations, collected, spanned and averaged recorded data graphs. Collected recorded data graphs plot data from different tests of the same set into one common coordinate system. They are used for the compact display of a large amount of recorded or computed test data with special emphasis on depicting the inter-test variance of the plotted quantity under retention of temporal resolution.

Averaged recorded data graphs serve the same purpose, however, they do not depict a series of curves but they average all corresponding curves and compute additional curves for upper and lower confidence boundaries. In contrast, the `SegmentRecordedDatafile` class, these two classes are not associated to individual tests but to wear test sets. Spanned recorded data graphs are useful to plot

curves from multiple segments of an individual test as one continuous line, see bottom left of Figure 33. The second group of graphs are those for aggregated data. They can be used to display dependent variables as a function of one independent variable, e.g. linear wear rate vs. sliding speed for multi-segment test with variable sliding speed. This type of plot is available at three aggregation levels: for individual plots, for test sets and for projects. Then, surface response graphs are plots of dependent variables as a function of two independent variables, e.g. coefficient of friction vs. loading and speed. This type of graph is available for the same aggregation levels than their single-parametric counterparts however here these three separate functionalities have been joined into one single class.

For plot generation duck typing is used: each class possesses a `plot` instance method which uses `gnuplot` as a backend [60]. These specialized plot methods implement the domain knowledge and rules of their respective classes, like knowing how to load or aggregate data according to the data source they are associated to (test, test set or project). Figure 33 shows a selection of graphs that can be plotted with *Atlas* built-in functionality. For cases in which a graph needs to be customized beyond the standard functionality, *Atlas* provides the option to download a graph, including the data file that contains the plotted data and the `gnuplot` script that was used to generate the graph.

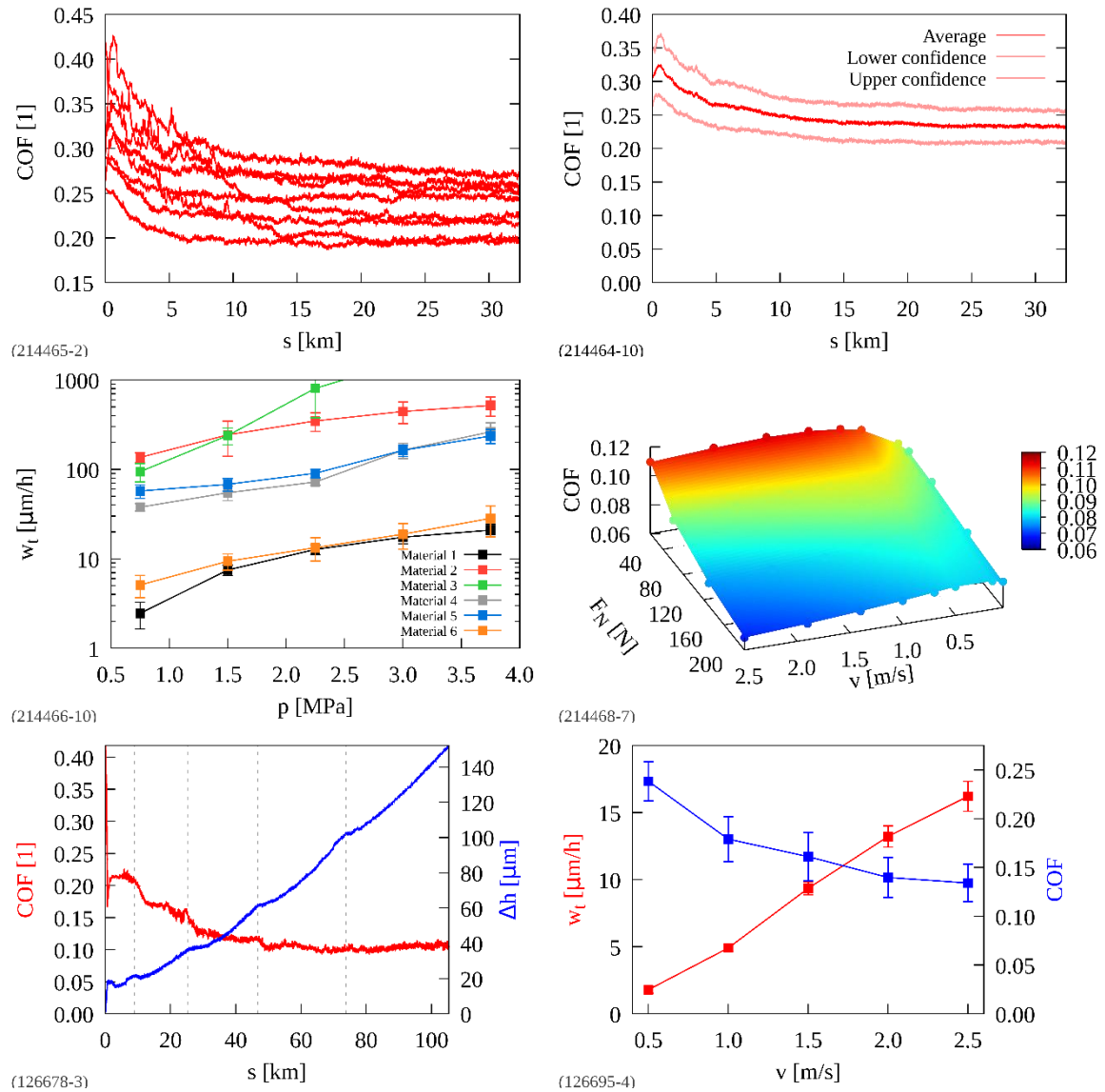


Figure 33: Exemplary plots. Top left: collection of the COFs of eight individual tests, top right: arithmetic mean, upper and lower confidence of the COFs of 8 individual tests, center left: linear wear rates of 6 different materials, center right: response surface plot (COF vs. normal force and sliding speed), data points represent arithmetic means of a set of 12 individual tests, bottom left: multi-segment of recorded data, bottom right: COF and linear wear rate versus sliding speed for a set of eight.

General test data		Test specimen data		Actions		Data files		Transfer film luminance analysis		Profilometer scans	
Project	644: DFG Transfer...	Block	Block 37501	Evaluate all segments	Name	MB		Image series is available. View	CTL	ITL	
Wear test set	1704	Material	EP-10SCF-8G-5SiO2 (F400)	Reset all segments	00550745.0001.dat	2.418		Re-evaluate luminosity data	post	post	
Status	DUE	Material	(1920)	Delete all datafiles	00550745.0001.orig	3.137			367	368	
Created at	06.05.2020, 15:51	W x G	4.0 x 4.0	Set to undone	00550745.0001.proc	0.169		Area (x,y)	Start (x,y)	Stop (x,y)	Size
Updated at	03.09.2020, 16:20	Ring lot	Ring lot 938	Set sliding distances from datafile	00550745.0001.xt	60.991		ROI	1026.823	727.910	299 x 13
Started at	06.05.2020, 15:53	Material	100Cr6 (1781)	Reset original sliding distances and units	00550745.0001.xt.header.yaml	0.001		REF	1216.823	1301.810	85 x 13
Tribometer	Atlas TT 1	ID x AD x W	50.0 x 60.0 x 25.0	Reset orig datafiles	00550745.0002.dat	1.912		POL	1026.748	727.740	299 x 8
Position	1			& correct initial stages	00550745.0002.orig	2.362		POLR	1216.748	1301.740	85 x 8
Operator	Bai Cheng Jim			Invert friction forces	00550745.0002.proc	0.168		Abs. ref. lum. corrected? No	ΔV	$\pm 0.007 \mu m$	n.a.
Angle	90.0°			Create Fickert01 physical aBoR	00550745.0002.xt	24.620		Correct $L_{abs, ref}$	ΔA	$\pm 0.006 mm^3$	n.a.
Eff. radius	30.0 mm			Create Mauerer physical aBoR	00550745.0002.xt.header.yaml	0.001		Polishing detection threshold (t) 108	R_a	n.a.	0.245
					00550745.0003.dat	3.261		Xt kinetics: 1 2 3	R_z	n.a.	1.809
					00550745.0003.orig	4.162					
					00550745.0003.proc	0.175					
					00550745.0003.xt	82.796					
					00550745.0003.xt.header.yaml	0.001					
					00550745.dat.header.yaml	0.002					
					00550745.log	0.004					
					00550745.orig.header.yaml	0.003					
					00550745.proc.header.yaml	0.004					
					00550745.span	0.139					
					00550745.span.header.yaml	0.001					
					00550745.zip	1.009					

Figure 34: Details page for an individual wear test. On this page, all relevant test parameters, associated objects (tribometer, test set, project) can be viewed. Furthermore, actions like evaluating or resetting one or all segments of a test can be executed, data files can be downloaded, plots of measured data can be viewed (not shown) and results and parameters of auxiliary investigations, like profilometer scans or transfer film luminance analysis (see Section 5) can be viewed.

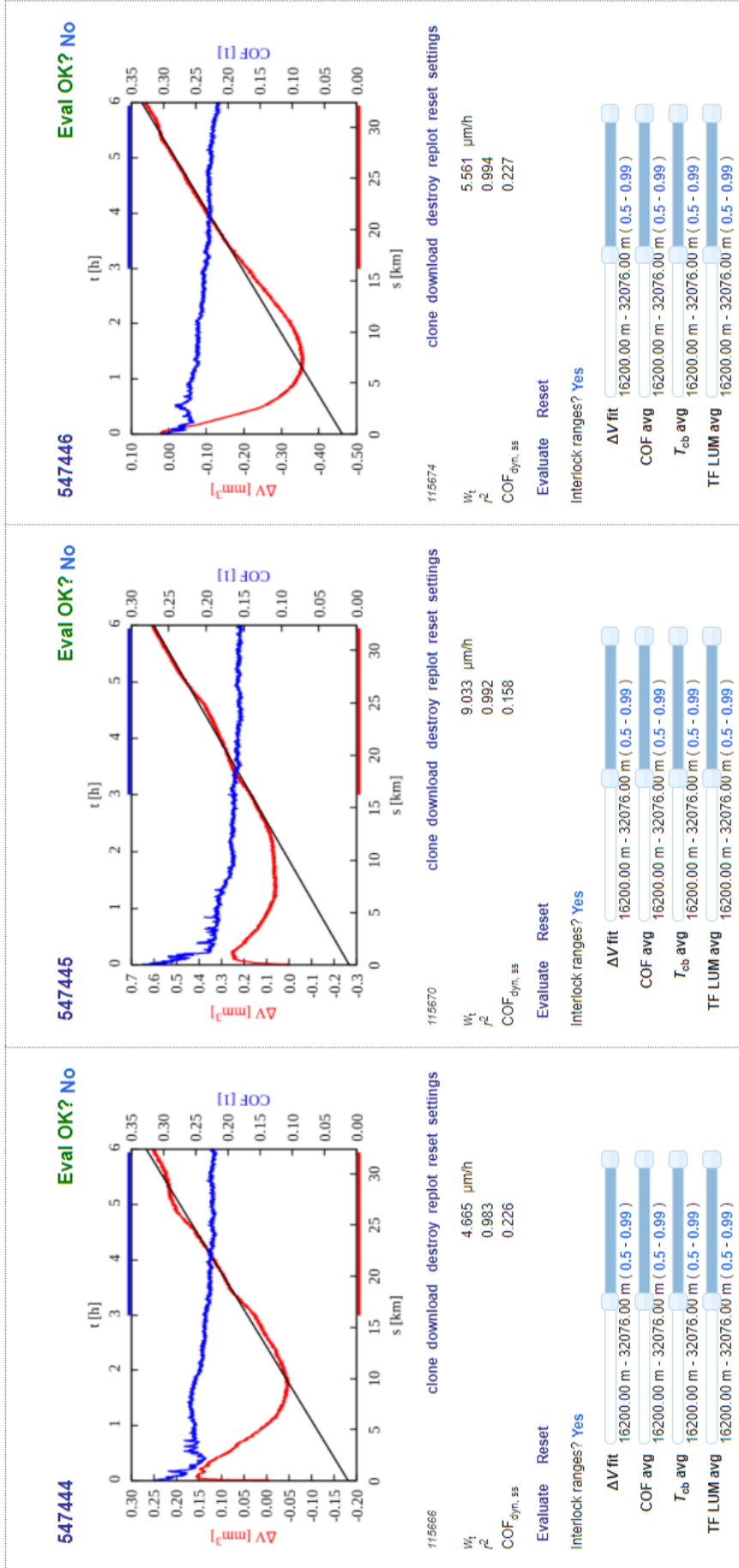


Figure 35: View for the joint evaluation of segments of different tests that belong to the same wear test set. All segments have the same segment index. This overview helps the user with assessing the validity of the evaluation of individual segments. The view offers control elements for adjusting the ranges in which various properties should be calculated. Furthermore, it provides basic results for each segment, like linear wear rate and coefficient of friction. Finally, it provides information about the overall validity status of the evaluation as well as control to set this status.

Test specimen 1	Test specimen 2	Set	Segment type	Loading	Sliding speed	Sliding distance	Duration	Tribometer	Wear test id	Test specimen ids	W_t $\mu\text{m/h}$	$W_{s,diff,ops}$ $10^{-6} \text{ mm}^3/\text{Nm}$	COF dyn,ss	ρ_{cb} [°C]	Status	Actions
TRIBOFORCE PA66 C0200 UNGEFAERBT (470) Lot 1311: 0000900743 TSC 676: no name TSL 942: no name	100Cr6 (1781) no material lot TSC 8: no name TSL 932: no name	1331 / 1	Slide	3.0 MPa	1.5 m/s	32400.0 m	06:00:00	Atlas TT 2	547444 33801 L932	547445 33802 L932 547446 33803 L932 547447 33804 L932 547448 33801 L932 547449 33802 L932 547450 33803 L932 547451 33804 L932	5 9 6 8 7 9 7	0.29 0.56 0.34 0.29 0.48 0.45 0.53 0.45	0.23 0.16 0.23 0.24 0.23 0.15 0.22 0.23	24.8 24.8 24.1 24.1 25.0 25.0 24.5 24.5	DUE A DUE A DUE A DUE A DUE A DUE A DUE A DUE A	E R E R E R E R E R E R E R E R
TRIBOFORCE PA66 C0200 UNGEFAERBT (470) Lot 1311: 0000900743 TSC 676: no name TSL 942: no name	100Cr6 (1781) no material lot TSC 8: no name TSL 932: no name	1336 / 1	Slide	3.0 MPa	1.5 m/s	32400.0 m	06:00:00	Atlas TT 2	547483 33889 L932	547484 33890 L932 547485 33891 L932 547486 33892 L932 547487 33889 L932 547488 33890 L932 547489 33891 L932 547490 33892 L932	6.9 4.8 7.5 5.7 4.4 7.3 5.0	0.42 0.30 0.46 0.35 0.27 0.34 0.45 0.31	0.18 0.15 0.18 0.19 0.18 0.14 0.20 0.20	24.9 24.9 24.5 24.5 25.3 25.3 24.8 24.8	DUE A DUE A DUE A DUE A DUE A DUE A DUE A DUE A	E R E R E R E R E R E R E R E R
TRIBOFORCE PA66 C0200 UNGEFAERBT (470) Lot 1311: 0000900743 TSC 676: no name TSL 942: no name	100Cr6 (1781) no material lot TSC 8: no name TSL 932: no name	1358 / 1	Slide	3.0 MPa	1.5 m/s	32400.0 m	06:00:00	Atlas TT 2	547635 33921 L932	547636 33922 L932 547637 33923 L932 547638 33924 L932 547639 33921 L932 547640 33922 L932 547641 33923 L932 547642 33924 L932	6.7 6.2 6.6 5.6 3.3 4.8 6.6 5.0	0.42 0.38 0.41 0.35 0.21 0.29 0.41 0.31	0.23 0.20 0.23 0.28 0.24 0.22 0.24 0.29	43 46 45 42 42 46 45 43	DUE A DUE A DUE A DUE A DUE A DUE A DUE A DUE A	E R E R E R E R E R E R E R E R
											5.9 ± 1.0 n = 8 s = 1.2	0.36 ± 0.06 n = 8 s = 0.07	0.17 ± 0.02 n = 8 s = 0.02	24.9 ± 0.3 n = 8 s = 0.3	E R L H B P h	
											5.9 ± 0.8 n = 7 s = 0.8	0.37 ± 0.05 n = 7 s = 0.05	0.24 ± 0.02 n = 8 s = 0.03	44 ± 1 n = 8 s = 2	E R L H B P h	

Figure 36: Project-level view of wear tests. Tests that belong to a common set are grouped together and groups are sorted by segment index in ascending order. In contrast to test-level and set-level view, no time resolved plots of measured or derived quantities are shown. Instead, test-level aggregates are shown. Typical examples for this are arithmetic means of various temperatures or the coefficient of friction. The list of displayed columns can be defined in a wide range and on a per-project basis. In addition to test-level aggregates, set-level aggregates are shown. These are mostly statistical data like arithmetic mean, standard deviation, confidence interval or the result of an automatic outlier test. In addition, validity data are shown and can be modified.

4.11 Assessment and storage of the validity of results

While convenience and consistency are important reasons for automated visualization of results, it also serves as a foundation for the assessment of the validity of results. This assessment is made on the level of individual test results, e.g. for COF, linear wear rate or measured test specimen temperature. *Atlas* supports two reasons for a result to be invalid: as the result of an automated outlier test (according to Nalimov's modification of Grubb's test [46]), or as the results of a user's decision. This decision is based on the user's academic training, experience and intuition. If a user classifies a result as invalid, this decision is stored to the database and supersedes any automatic outlier test. Any invalid results will be excluded from automatic calculations, like that of the arithmetic mean of a set of results that belong to a given test set. Furthermore, it is ignored when data is aggregated for plotting aggregated data graphs.

Atlas provides the possibility of classifying a result as invalid only for a limited set of results, amongst which are steady state COF, linear, specific and gravimetric wear rate, mass loss, static COF, frictional power density or measured test specimen temperatures. As these results are persisted on the level of individual wear test segments, *Atlas* implements the instance method `valid` for the `WtSegment` class, see Code listing 28.

Code listing 28: Implementation of the `valid` method on the test segment-level.

```
class WtSegment < ApplicationRecord
  def valid(attribute)
    case attribute.to_sym
    when :ambient_humidity
      return ambient_humidity_data_valid
    when :ambient_temperature
      return ambient_temperature_data_valid
    # more branches for more attributes that can be invalid
    end
  end
end
```

This method consists of a switch-statement in whose branches the algorithms are implemented according to which the decision is made whether the requested attribute is valid or invalid. In the shown cases of ambient humidity and temperature, the decision is simply made by calling for corresponding attributes of the segment

in question. These are stored as Booleans and therefore do not require any additional processing or type casting. All other branches are also programmed to return Boolean values. Therefore, the return value of this method is used for the mentioned exclusion from calculations or from data aggregation.

Furthermore, in combination with the automatically conducted Nalimov's test, it is used to display the status of a given result in several of *Atlas*' result views.

4.12 Automatic report generation

In addition to its in-browser views, *Atlas* can also render reports in PDF format. While the details of the implementation of this process, which basically is rendering the same information, or a subset thereof in a different output format, is beyond the scope of this document. However, Figure 37 shows the form that is provided for customizing the content of such a report and a full report is shown in the Appendix.

Generate PDF report

Inclusions

Recorded data graph
y: ▼
y2: ▼

ΔL_{rel}
 xt map

Attachments
 Wear scar scans
 Test specimen table

Pre-processing

Harmonize recorded data graphs

Disposition

Download
 View in Browser

Include statistics with outliers?

Yes
 No

Reported results

w_t [$\mu\text{m}/\text{h}$]
 w [$\mu\text{m}/\text{km}$]

$w_{s, \text{diff, ops}}$
 $w_{s, \text{int, topo}}$
 $\Delta V_{\text{int, topo}}$
 $\text{COF}_{\text{dyn, ss}}$
 $\text{COF}_{\text{static}}$
 $\vartheta_{\text{cb, avg}}$
 $\Delta L_{rel, avg}$
 Δh_{ops} (100 %)
 Δm

Additional significant digits

(0 = auto)

Language

English
 German

Figure 37: Form for customizing the content of automatically generated reports.

4.13 Limitations and known issues

Authentication only: Out of the standard AAAA-concepts for LIMS (Authentication, Authorization, Auditing, Accountability), *Atlas* only implements authentication.

Units of physical quantities: Although *Atlas* has a system for handling units of physical quantities, see Code listing 24, it does not use it on all possible occasions, see Figure 11 for an example of hardcoded quantity units (gauge in mm).

Only flat material composition: Particle-filled polymers are a popular class of materials in tribology. Frequently, a single material consists of 2-8 different fillers, each of which having a different type, brand name, particle size distribution and filler content. The “filler” attribute that *Atlas* includes for its material model is however only a single-valued string attribute. This issue could be solved by, developing the material model into a composite model whose components contain information about names and the content of the individual components, i.e. matrix and fillers.

No material processing: *Atlas* does not contain any data models for the provenance of test specimens or for processing information on materials.

Limited interoperability: *Atlas* uses auto-incremented sequences of integers as primary keys for all tables. These sequences are maintained locally on each server instance. Specifically, no coordination happens between individual server instances which greatly impairs the exchange of records between *Atlas* server instances.

No support for measuring equipment monitoring: In its current implementation, *Atlas* does not provide a subsystem for managing measuring equipment.

Limited code (quality) management: While *Atlas*' application code is subject to version control and bug-tracking using gitlab, it is not subject to automated testing (e.g. using test-driven development) or formal specification (e.g. using RSpec).

Limited security: *Atlas* has been designed for use in a protected and trusted environment. It implements some practices which are considered security risks: it does not require SSL encryption, does not limit session duration or makes use of security features provided by Rails, like cookie rotation or strong parameters [59].

5 Transfer Film Luminance Analysis

5.1 State of the Art

Polymers and polymeric composites are in widespread use for tribological applications in general and especially for sliding applications. An elementary process of sliding is the formation of a so-called 'transfer films' which is the (mutual) deposition of worn material on the other sliding partner. The mechanisms by which transferred material resists its potential removal are considered to be adhesion, mechanical interlocking and by the formation of covalent bonds [61]. Despite being called 'films', microscopic inspection of transfer films often reveals that they are actually rather a collection of local material depositions. In the most extreme case, they can consist of only minimal amounts of debris that only just fill the valleys of the roughness profile [62]. However, due to the limited optical magnification of the human eye they typically appear as coherent material layers.

Transfer films formed from worn polymeric material were reported as early as 1964 by Makinson and Tabor who identified the formation of a polymeric film made of abraded polytetrafluorethylene (PTFE) on glass [63]. The authors reported a coincidence of the occurrence of high coefficients of friction (>0.3) and 'lumpy' transfer films, while coefficients of friction between 0.07 and 0.3 coincided with homogeneously distributed films. This observation, i.e. a high variability in the morphology of formed transfer films as well as the accompanying variable impact on friction, was also reported by Lancaster, however this time concerning the impact of transfer films on wear [64], [65]. He found that transfer films can smoothen the metallic surface and thus lead to lower localized stress and therefore to lower wear. However, Lancaster also reported an increase of wear due to transfer film formation, especially in the case of brittle materials such as epoxy resins or brittle thermoplastics like polystyrene and polyester. From these contradictory findings, Lancaster concluded that the influence of transferred material on friction and wear would be a material property.

In 1978 Rhee and Ludema reported that the formation of a transfer film of PA66/PTFE on steel increased the temperature at which mild wear transitions to severe wear is increased by 50 °C [66]. In 1991 Rhee et al reported that transfer films not only affect friction and wear, but also the noise production in automobile

brakes [67]. In 2015, Zhang et al. found that transfer films can also prevent tribologically induced oxidation [68]. Today, the formation of a transfer film is considered to be one of the reasons for the typical observation of a "gradual transition from a transient wear behavior to steady state wear behavior" [69].

Recently, Alam et al. reported on the importance of stable transfer films for wear rates of PTFE-based composites to be 'ultra-low' ($< 10^{-7}$ mm³/Nm) [70]. Furthermore, the stability of transfer films formed by MoS₂ has been identified by Yu et al. as prerequisite for "superlow" friction of H-DLC films [71]. For brake pads, Nishimura et al. found that the stability of transfer films formed by copper-free materials directly impacted the temporal stability of the coefficient of friction and that transfer film stability was influenced by material composition [72]. Similar findings were made for asbestos-free brake pad materials [73].

The significant impact of transfer films on the behavior of tribological systems has prompted many studies on transfer films, specifically on their detection, assessment and quantification. As a result, a wide set of methods has been established for these purposes.

Microscopic inspection is the family of detection methods for transfer films that is most frequently used. Its most prominent members are optical microscopy (OM), scanning electron microscopy (SEM) [61], transmission electron microscopy (TEM) [74] and scanning transmission electron microscopy (STEM) [75]. Typically, the main purpose of microscopy is the visual assessment of the transfer film, e.g. with respect to the shape, size and distribution of the deposited material with high to extreme spatial resolution. Electron microscopy is frequently combined with focused ion beam (FIB) for in-situ preparation of cross sections for the quantification of the (local) film height and with energy dispersive x-ray analysis (EDX) for the elucidation of the chemical elements of the transferred material.

In 2014 Ye et al. reported on an effort to amend the issue of non-numerical results from microscopy: they introduced "free-space length" which is a quantitative measure for blank regions in-between patches of transferred material, based on optical micrographs, SEM images or profilometric data [76]. The authors were able to demonstrate the coincidence of the reduction of the free-space length and the specific wear rate of PTFE sliding on 440C steel. Despite the obvious usefulness of their approach, they were only able to solve the issue of the high locality of their analysis only by recording multiple OM images and by then stitching them together.

It is therefore probably no coincidence that they selected an oscillation experiment with a stroke of only 25.4 mm as to keep the effort at a reasonable level. With today's sophisticated microscopes with automatic multi-imaging routines and algorithms for auto-stitching, capturing an image of the full wear track is possibly feasible for flat specimen, e.g. discs from pin on disc tests. However, for non-flat wear tracks, e.g. on a ring, a shaft or a gear tooth, the test specimen must not be translated laterally but rotated in order to capture the whole wear track. However, neither auto-rotatory specimen holders nor image recording software that is synchronizable with such a device is standard tribology laboratory equipment.

Apart from issues with image capturing, the process of determining the free-space length requires the user to provide the algorithm with training data for each image. Therefore, the algorithm must be provided with a set of user-defined and user-classified regions in each image. Also, the procedure is *ex-situ*. Therefore, in order to achieve time resolved data, the test must be operated intermittently, with all the usual implications for its result (as friction and wear are no material properties but instead depend on system history).

Despite being undoubtedly useful, microscopic techniques are typically *ex-situ*, highly localized and require expensive equipment (more so in the case of S(T)EM/EDX/FIB, less so in case of OM) and they hardly yield quantitative data that can be directly correlated to other test data, e.g. with coefficient of friction or wear data. This is true regardless of whether this other data is available as time-resolved *in-situ* data series, as intermittent *ex-situ* data or only in the form of the difference between initial and final state (e.g. from difference weighting). Last but not least, its highly local nature makes microscopic analyses susceptible to a generalization error: local observations are used to explain other observed phenomena, e.g. wear or friction data which, however, are the integral result of vast number of local phenomena and processes. While an experienced and responsible researcher might be able to make such a generalization, it is still very hard for the reader to re-examine the drawn conclusions, as typically only a pre-selected set of local observations is presented.

Surface metrology methods include optical methods like laser or white light profilometry and tactile methods like atomic force microscopy (AFM) [77]. As modern metrological methods reach vertical resolutions of single digit nanometers and below, their main goal is typically to directly measure the height of the transfer film.

While optical methods are basically suitable to record laterally extensive images – in the best of cases even images of the whole wear track -- tactile methods are typically restricted in the size of the region of interest they can record. Restrictions can be by design, e.g. being unable to scan a wear track on a ring without an auto-rotator, or they can be practical restrictions, like uneconomically long measurement times. Furthermore, for 'thin transfer films', i.e. films that only consist of wear debris in the valleys of surface profile, it can be difficult to determine transfer film height at all, as it potentially is smaller than the height variations of the surface profile.

Profilometry is another sub-family of surface metrology and it aims at measuring the surface profile of a worn surface with such a high resolution that the height of the transfer film can be read from the profile height data directly. One of the many studies that utilizes this technique is the one of Li et al from 2015 in which they found a correlation of the transfer film height, the initial surface topography and friction and wear [78].

In 2014, Sebastian implemented an in-situ measurement of the transfer film height using a white light displacement sensor on a custom-built block-on-ring tribometer [79]. It yields time-resolved in-situ data of the transfer film height that is synchronous with all other in-situ measured data. It is therefore, by design, very well-suited to understand the dynamics of transfer film formation and the impact it has on friction and wear. However, until today it has not yet found wide-spread use.

Overall, although this family of methods basically yields quantitative data, like microscopy, they are also mostly ex-situ only. Additionally, although sometimes possible in theory, in practice it is often so costly to produce laterally extensive measurements of the wear track that researchers often resort to local investigations, thereby running the risk of the mentioned generalization error.

Micro-mechanical analysis is a group of methods that does not directly aim at measuring the transfer film height. Instead, they focus on collecting information about mechanical properties of the film. Most often, the hardness of the film is measured. As transfer film 'heights' often are below 1 μm , mandates the use of nano-indentation as to avoid substrate artifacts. Exemplary, such investigations were made by Randall [80], [81] and Chang et al [77]. Zhang et al used the differences in elastic moduli of transfer film and substrate to produce a transfer film coverage map [68].

Overall, micro-mechanical analysis techniques typically are ex-situ, highly local and time consuming. Therefore, all the above-mentioned issues apply. Furthermore, their results, despite being mostly quantitative are even harder to correlate with friction forces or wear rates as there is no standard model which easily relates transfer film hardness or electric resistivity with either.

Spectroscopy basically yields information of the wavelength dependent interaction of light (or more general: electromagnetic waves) with matter. It is a highly developed and specialized field of natural science and has been used extensively to study transfer films. Therefore, only the most important methods, with respect to the above formulated criteria, are cited. Infrared (IR) spectroscopy has been used in many ways for investigating transfer films. Jain and Bahadur used quantitative transmission absorption IR spectroscopy to directly measure the thickness of transfer films [82]. However, their method is only applicable to situations on which both test specimen do not absorb IR radiation completely. As some absorption is needed by design, and as the optical absorption coefficients of solid are typically very high, tests must be made with thin films which are difficult to prepare, handle and test.

In 2003, Scharf and Singer introduced an in-situ Raman technique that enables the measurement of the transfer film height by ex-situ calibration of Raman spectra to film heights and coverage ratios [83], [84]. Using this technique, they were able to demonstrate the temporal coincidence of transfer film build-up and reduction of the coefficient of friction during the running-in phase of an experiment as well as the temporal coincidence of transfer film degradation and fluctuations of the coefficient of friction in later stages of the experiment. While this technique yields quantitative and time resolved data about the transfer film thickness and about the percentage of the wear track that is covered with transfer film, it also possesses some serious issues. First, the used infrared radiation needs to pass through one of the two sliding partners which seriously restricts the material combinations that can be investigated. Second, as the initial transparency of the test specimen, typically a glass or sapphire hemisphere, degrades over the course of an experiment (due to wear) and as loose debris accumulates in the observed region of interest, this technique can only yield data for a very limited amount of time.

In 2008, Singer et al reported on an in-situ Raman IR analysis that was able to detect MoS₂ which has been formed by tribomutation when sliding glass on a Mo-

S-Pb anti-friction coating and which they identified as the reason for low friction and wear [85]. Although the authors concluded that their technique "... opens a new window on the buried sliding interface", as of today it has not yet found widespread use, most likely due to the required highly specialized equipment (a Raman spectrometer converted to a tribometer).

Photo-optical analysis includes all techniques that directly observe the wear track through photo- or videographic instruments. Images can be recorded continually or intermittently and can be evaluated either numerically or by visual inspection.

In 2007, Chromik et al. introduced the so-called "Newtonian interference analysis", also known as Newton's ring analysis, [86], [87]. By monitoring the relative position of Newton's rings outside the contact zone as sliding progresses, the thickness of transfer films can be quantified. Comparable to the in-situ Raman spectroscopy introduced by Scharf and Singer [84], this method also yields quantitative, laterally extensive and in-situ data. Still, in requiring at least one of the two sliding partners to be transparent, the selection of material pairings is also seriously restricted. Furthermore, the non-transparent test specimen needs to be very smooth, i.e. polished which restricts the range of simulatable applications even further. Also, the accuracy of the measurement degrades over the course of an experiment if the initially polished surface gets roughened during the test.

In 2016, Meneghetti et al. introduced the continual video observation and greyscale analysis for a twin disc tribometer [88]. They did not explicitly study transfer films but instead used the detection of certain above-threshold greyscale values as an indicator for the onset of pitting. Although not aimed at the detection, analysis or quantification of transfer film, the method is a prime example for how continual optical in-situ inspection can be used to produce time-resolved, quantitative and laterally extensive data.

Autoradiographic transfer film analysis is based on the spatially resolved detection of radioactive decay. This requires one of the two sliding partners to contain a radioactive nuclide that is transferred to the other, non-radioactive specimen during transfer film formation. Therefore, at whatever position the non-activated specimen emits an elevated level of radioactivity, it must be covered by transferred material. Applying this technique to dry metal/metal sliding, Kerridge and Lancaster were able to demonstrate the build-up of a transfer film. However, as the method could

not be applied in-situ, temporal resolution was only achieved by intermittent autoradiography [89]. Rabinowicz and Tabor reported a correlation between the amount of transferred metal and the observed coefficient of friction [90], [91].

The major issue of this method is that it requires a chemical element that can be activated by radiation in order to be used as a marker. In contrast to metal/metal sliding contacts, polymers however do not typically include atoms that can be easily activated.

5.2 Shortcomings of the state of the art, objectives and solution procedure

Overall, the presented selection of existing methods for observing and analyzing transfer films exhibit five major shortcomings:

1. Data cannot be collected in-situ and thus temporal resolution can only be achieved by intermittent test operation, (potentially) including the temporary removal of the test specimen from the tribometer.
2. The analysis is highly local, either by principle or due to time and costs restrictions.
3. The collected data is either non-quantitative by design or quantitative data is only obtainable through non-trivial and time-consuming calibration.
4. The selection of materials and test specimen designs to which the method can be applied is restricted by design, e.g. by requiring transparency or radioactive emission. Often these restrictions are severe.
5. The required testing equipment is expensive, very special-purpose, requires special testing environments (e.g. vacuum) or is not even commercially available and/or requires highly trained personnel.

Table 5 classifies a selection of existing methods according to these five categories. For comparison, the novel photo-optical luminance analysis is also included. While recently no significant progress has been made in terms of methodology, researchers have used the existing methods to investigate transfer films. In each of their reports, one or more the identified shortcomings can be readily identified.

Table 5: Classification of methods for transfer film analysis based on five selected issues.

Method	Quantitative Quantity	In-situ Sustainable	Laterally Extensive	Economic Efficiency	Material Selection
Microscopy					
- OM	No	No	No	Medium	Unrestricted
- OM + Free space length	Yes, free space length	No	No	Low	Unrestricted
- SEM	No	No	No	Low	Unrestricted
- SEM + FIB	Yes, film height	No	No	Very low	Unrestricted
- TEM	Yes, film height	No	No	Very low	Unrestricted
Surface metrology					
- Free space length	Yes, free space length	No	Possible	Medium	Unrestricted
- Profilometry	Yes, film height	No	Possible	Medium	Unrestricted
Spectroscopy					
- X-ray spectroscopy	No	No	No	Low	Unrestricted
- Inelastic Raman scattering	Yes, film height	Yes, No	Yes	Low	Transparent
- Quant. Infrared spectroscopy	Yes, film height	No	No	Low	Transparent
Photo-Optical					
- Newtonian interference analys.	Yes, film height	Yes, No	Yes	Low	Transparent
- Luminance analysis*	Yes, luminance change	Yes, Yes	Yes	High	Unrestricted
Autoradiography	No	No	No	Low	Radioactive

* novel method presented herein

In 2019, Placette et al. have studied the impact of relative orientation between sliding direction and grinding direction on the formation of transfer films by neat PEEK in dry sliding against hardened A36 steel [92]. They utilized a white light profilometer to determine film volume and height and the fraction of the wear track area that was covered with transfer film. In order to reach quasi-lateral extensivity of their data, they sampled the wear track in discrete intervals and then extrapolated to the full wear track area. Furthermore, Raman IR spectroscopy was used locally to determine the chemical composition of the films. All investigations on formed transfer films were done exclusively ex-situ. Therefore, no time-resolved data could be obtained. Although the authors state that they used white light interferometry to de-

termine transfer film volume and height, no numerical data from these investigations are presented. From purely reciprocating tests, they found that transfer film formation was less pronounced in parallel than in perpendicular orientation. On average, more wear was found for parallel orientation and hence the authors concluded that the formation of less transfer film was the reason for increased "potential for wear". They reported on a series of other findings as well, however, as they only did 1-3 individual tests per testing condition, i.e. combination of wear path (square or linear reciprocation) and sliding speed, and taking into account the significant variation that is deferrable from the presented graphs, it is highly questionable whether any of their conclusions would hold up to a statistical hypothesis test. Also in 2019, Bashandeh et al. investigated the usefulness of graphene nano platelets and PTFE as solid lubricants for polymeric coatings at temperatures up to 300 °C. [93]. From their experiments, they found that transfer films had been formed. The investigation methods were optical and scanning electron microscopy. Again, all investigations were done post-test only and only qualitative results were obtained by interpreting the taken images. The authors found that temperature had a significant effect on the extent of transfer film formation, with more material being transferred with increasing temperature. Furthermore, the authors state for coatings with graphene nanoplatelets "a more uniform and continuous transferred film was developed with increase of temperature". The evidence provided is a series of OM images from which the reader can either agree with the author's qualitative findings or not. In no way could he verify that the presented images are either representative for the whole wear tracks or reproducible over a series of experiments. In 2020, Sun et al. reported on the ultralow wear of PTFE-based composites filled with beryllia and germania particles [94]. In this study wear tests were run intermittently in order to produce time-resolved wear data. Transfer films were investigated using IR spectroscopy, OM, laser microscopy and stylus profilometry. From stylus profilometry transfer film height was calculated. In the case of IR spectroscopy, the authors commented on how they tried to produce laterally extensive data: "Spectra were collected at five different locations and averaged to obtain a single representative." In the case of stylus profilometry, however, this aspect of data acquisition remains without comment. Furthermore, it is noteworthy that for microscopy, stylus profilometry and spectroscopy only post-test results are presented. Still, although not having suitable experimental data -- i.e. time-resolved and quantitative

data on the extent of transfer film formation, to support such a hypothesis, the authors wrote that the observed transition from severe to mild wear of BeO- and GeO₂-filled PTFE "[...] reflects the development of wear resistant tribofilms at both the sliding surfaces [...]".

Most recently, Cui et al. investigated the role of transfer film formation on the tribological properties of composite materials with a PEEK/PTFE hybrid matrix and additional fillers like carbon fibres, molybdenum disulfide, graphite and graphene nanoplates sliding against "GGr15" steel in vacuum at temperatures from 20 to -100 °C [95]. In this study, transfer film morphology was characterized using SEM and confocal Raman microscopy and transfer film chemical composition was investigated using energy dispersive spectrometry (EDS) and infrared spectroscopy. All these investigations were done after the tests were finished, i.e. post-test. Among other, the authors found that at temperatures -30 °C and below, transfer films were formed on the steel counter bodies. Furthermore, they attributed an observed fluctuation of the coefficient of friction to an instability of the formed transfer film. Specifically, they wrote: "When the temperature is -30 °C, the formation and removal of transfer film occur simultaneously, leading to the instability of friction coefficient in the early stage of sliding, which finally stabilized at high friction due to the failure of transfer film formation as evidenced by FTIR characterization." However, given that all their transfer film investigations took only place after the end of their tests, it remains unclear how exactly it was conceived that the observed fluctuations of the coefficient of friction actually coincided temporally with instabilities of the transfer film. Besides this, it remains unclear which physical quantity would have been chosen to represent transfer film formation numerically and how corresponding numerical data would have been obtained in a time-resolved manner.

Overall, even from this very limited selection of recent studies on transfer film research, it becomes obvious that studies typically do not include the production of time-resolved and quantitative data on transfer film formation. Furthermore, the issue of lateral extensiveness is not always properly addressed. Still, it happens that researchers relate time-dependent friction and wear data to the temporal evolution of transfer film formation without clarifying how exactly these conclusions are based on either time-resolved data or how they could have been retro-polated from post-test investigations.

Based on the shortcomings of the state of the art on in-situ time-resolved transfer film detection and quantification and on the demonstrated shortfalls that the lack of such information is regularly producing in the research on transfer films, the second aim of this thesis was to develop a novel method for detecting transfer films. This method should solve as many of the identified issues as possible. Specifically, the acquired data should be quantitative, in-situ, laterally extensive, universally applicable (in terms of test setup, material selection and test specimen shape and surface roughness), infinitely sustainable, require only components that are commercially available and affordable and should not require advanced personnel training. Finally, the primary data produced by the method should be stored in the presented LIMS and data processing, data plotting and aggregated data storage should also be handled by the presented LIMS.

5.3 Photo-optical transfer film luminance analysis (TLA)

5.3.1 Preliminary considerations and assumptions

Basically, transfer films can be formed on both sliding partners. In polymer metal-slide pairings, however, transfer films are usually formed exclusively 'one-way', i.e. by the transfer of worn polymeric material to the metallic sliding partner. The method and apparatus presented in this study therefore describe the observation and measurement of such a polymer-based transfer film on a metallic sliding partner. However, with some simple adaptations and device duplication, it can also be applied to situations where transfer films are potentially formed on both materials. Furthermore, for ease of use, the simple geometric setup of an ASTM G137 block-on-ring test has been utilized for demonstrating this method for the very first time.

5.3.2 Measurement principle, test setup and data recording

The fundamental principle of the presented method is to measure the change of the wear track luminance on the metallic test specimen. It is assumed that changes in luminance result from the absorption of light by the formation of a transfer film during the experiment. Luminance is the photometric measure for the amount of visible light which an illuminated body reflects into a given direction.

The data acquisition is therefore based on the capturing of photographic images of the wear track on the metallic counter body, in this case a steel ring, during the

whole test. Typically, images are taken in regular intervals whose length is chosen by the experimenter. One of the main requirements for taking the images is that the lighting situation, i.e. the intensity and the angular distribution of incident light, should remain constant throughout an experiment. This is best achieved by the utilization of a dome lighting.

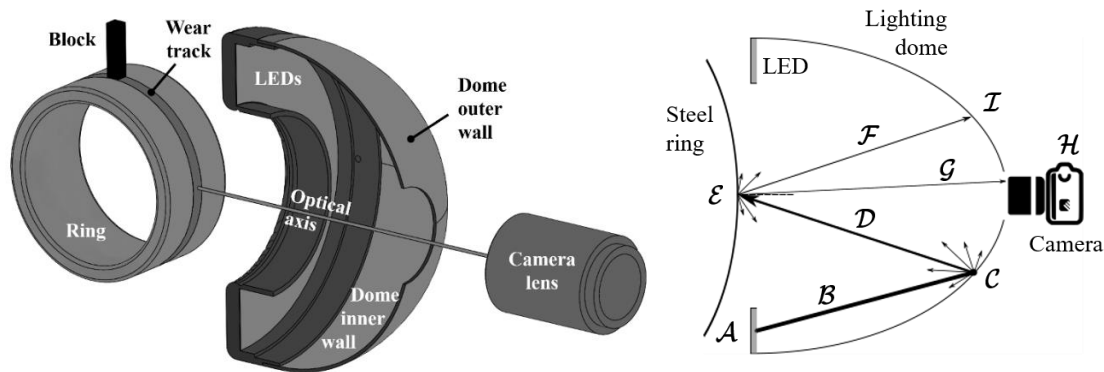


Figure 38: Schematics of the experimental setup for a block on ring experiment: diagonal view with longitudinal section through the light dome (left) and section of the whole test setup showing typical ray traces (right)

Figure 36 shows a schematic of the test setup, including the position of the lighting and the camera with respect to the ring. It is crucial that the center hole of the dome, the camera lens and the radius of the ring are aligned. During the test, (white) light is emitted from light emitting diodes that are integrated into the base of the light dome (A). As an example – there are many more ray paths not shown here – a light ray B is shown. It hits the inner surface of the light dome, which is painted in matte white, at point C where it gets reflected diffusely. Again, out of the many ray traces that start at this point, ray D is shown as an example. It travels to the steel ring and hits it at point E . The actual reflection from the metal surface can be viewed as a superposition of perfect specular reflection and perfect diffuse reflection, i.e. mixed reflection. It is well known that the relative contributions of these two modes to actual reflection significantly depend on the surface roughness of the steel ring. Unless otherwise noted, diffuse reflection is assumed to be the predominant reflection mechanism. This is indicated by the emission of many rays of weak intensity (represented by thin lines) in many different directions. Some of these travel to the camera lens (G) and therefore ultimately reach the camera's light sensor. Ray F is an example for such a ray trace.

Although it is possible to do this test under ambient lighting conditions, it is recommended to isolate the experimental setup from ambient stray light at all times during the test.

In contrast to classical microscopic analysis of transfer films, the optical magnification of this setup is significantly lower. While sacrificing the ability to resolve individual patches of deposited wear debris, and more so to visualize their morphology, the aim is to observe the full lateral extent of the wear track. Figure 39 visualizes this strategy:

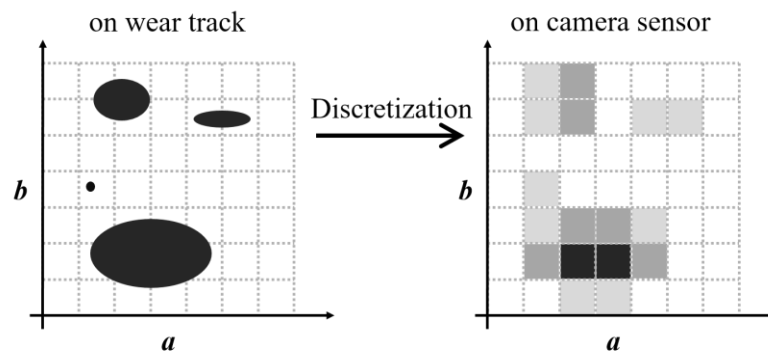


Figure 39: Wear debris patches located on the wear track are discretized on the camera sensor.

Instead of applying a high magnification on the four exemplarily depicted patches of transferred material – a strategy which ultimately is responsible for the high localization of classical micrographs – the optical system of the camera depicts the wear track with a magnification of approx. 1 (in this case) onto the camera's sensor. Due to the finite size of the sensor's pixels and due to the ratio of pixel size to patch size, the contours of the individual patches are blurred due to discretization.

While being exposed to light, each pixel registers photons that are emitted from a specific surface area increment of the ring in general and therefore also from the wear track. When photons hit the sensor at a position that represents a pixel of the recorded image at image position (a, b) with a being the horizontal position of the pixel and b being its vertical position, they are converted into an integer value. When using a color camera, actually three integers are recorded for each sensor position; one for each of the fundamental colors (red, green and blue). However, as color is of no relevance for this method, cameras should be operated in grayscale mode. Then, only one single pixel value is yielded according to:

$$N(a, b) = K \cdot \frac{t_{\text{exp}} \cdot S}{f^2} \cdot l(a, b) \quad (1)$$

where K is a calibration constant of the camera sensor, t_{exp} is the exposure time, S is the ISO sensitivity, f is the aperture number and $l(a,b)$ is the luminance of the area on the ring that the camera's lens (system) projects onto the sensor at a position that corresponds to image position (a,b) .

While a photograph of the ring already depicts a significantly larger portion of the whole wear track than any microscopic technique, a standard short exposure image does not yet contain information on the *whole* wear track. However, as the ring rotates during an actual wear test, the whole wear track can be easily captured in any individual image by setting the exposure time t_{exp} of the camera to an integral multiple n of the rotation period t_{rot} of the ring:

$$t_{\text{exp}} = n \cdot t_{\text{rot}} \quad (2)$$

Now, each circumferential position of the ring surface gets into the view of the camera once (more generally n times) during a single exposure. Therefore, the deposition of transferred material which is assumed to be encoded in the pixel values of the image is included into each picture. This way, the issue of many of the existing transfer film recording techniques of observing only a tiny fraction of the whole wear track, and thus of the whole transfer film, is solved. In order to discern them from short exposure images, images that satisfy the exposure time matching criterion of equation (2) are called "circumferentially averaged". Unless stated specifically otherwise, the term 'image' will always refer to a circumferentially averaged image.

Figure 40 shows a schematic of how circumferential averaging produces vertically averaged images while retaining the lateral resolution of the transfer film information contained in the image. Also, by the example of a short exposure image and a circumferentially averaged image of the *same* transfer film, it shows how the process of projecting each position along the whole circumference onto the sensor can change the brightness of the wear track at the marked position significantly. The reason for this is that (at any given point in time) the transfer film is not only inhomogeneously distributed in lateral direction but also in circumferential direction. Therefore, when rotating the ring, out-of-view circumferential sections with different transfer film coverages get into view of the camera. This leads to an averaging of the circumferential luminance distribution of the wear track. At the marked image area, this leads to a significant increase in brightness, because the short exposure

image has – by chance – recorded a local minimum in ring surface luminance. This is a good example for how highly local observations can yield non-representative results when the investigated transfer film is laterally highly inhomogeneous. The same concern applies to ex-situ, i.e. intermittent or post-test transfer film investigations where only a transient or the final state of a transfer film is investigated but where the findings are then used to explain friction and wear data that have been recorded at previous points in time without there being any guarantee that there is any mechanistic correlation between the data and the observed state of the transfer film.

Because of circumferential averaging, each pixel with the same a -position accumulates photons from a cylindrical section of the ring's surface which spans the whole circumference of the ring. Therefore, at any given a -position, the luminance of a circumferentially averaged image contains the same information on transfer film formation at all corresponding b -positions. The only variation of luminance in b -direction now originates from the curvature of the ring (which causes changes in the ratio of specular to diffuse reflection as a function of b -position).

5.3.3 Image processing and data extraction

At this point the concept of image segmentation is introduced which defines a region of interest (roi) and a reference area (ref). Figure 40 shows an example for such a segmentation.

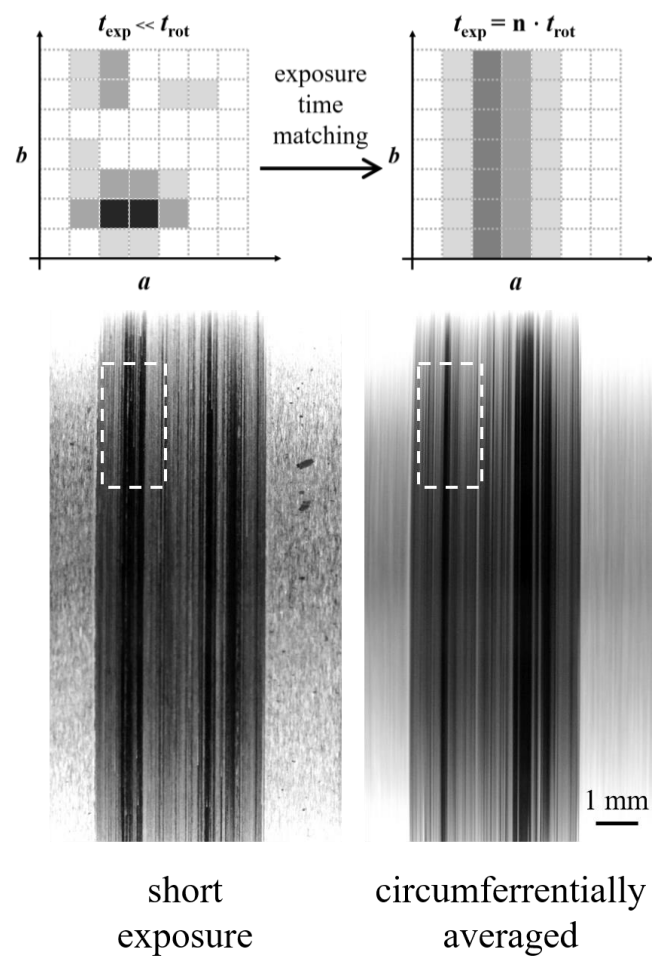


Figure 40: Transition from short exposure image to circumferentially averaged image by exposure time matching (top). Luminance change due to circumferential averaging (bottom).

To do so, a new coordinate system for the roi in which the horizontal in-roi pixel position A is defined as

$$A = a - a_i, \quad (3)$$

where a is the absolute horizontal pixel position with respect to the origin of the total image's coordinate system (in the lower left corner) and a_i is the roi's lateral starting point (in absolute image coordinates). Correspondingly, the vertical in-roi pixel position B is defined as

$$B = b - b_i. \quad (4)$$

Analogously, the pixel positions at which the roi ends in horizontal and vertical directions are called a_f and b_f respectively. Using these new quantities, roi width ΔA and roi height ΔB are defined as:

$$\Delta A = a_f - a_i. \quad (5)$$

$$\Delta B = b_f - b_i.$$

Using an asterisk to denote the reference area, analogous quantities are defined for the reference area:

$$A^* = a^* - a_i^* \quad (6)$$

$$B^* = b^* - b_i^*$$

$$\Delta A^* = a_f^* - a_i^*$$

$$\Delta B^* = b_f^* - b_i^*$$

The roi should typically span the whole width of the wear track. The reference area should be chosen as to that it exhibits lighting conditions equivalent to the roi. This is considered to be the case if

- the incident light has the same intensity and angular distribution in both roi and the reference area,
- and the ring has the same curvature in the both areas.

Strictly this is impossible due to the combination of the gradient in light intensity in vertical direction (caused by the ring's curvature) and the circular shape of the light dome which causes a light intensity gradient in both horizontal and vertical direction. In practice, roi and ref can be considered to be lighted equivalently if the following conditions apply:

- the dimensions of roi and ref are small compared to those of the ring and the light dome (in this case this ratio is smaller than 1:15)

- roi and ref begin and end at (pairwise) identical b -pixel positions (to ensure comparable curvatures of the ring), i.e. $b_i^* = b_i$ and $b_f^* = b_f$ and therefore $\Delta B^* = \Delta B$ (i.e. roi and ref are of equal height)
- roi and ref are in close proximity in a -direction (to ensure negligible changes in incident intensity due to non-zero lateral intensity gradients)
- images should strictly be recorded in radial direction and the camera should look at the ring in normal direction
- roi and ref should be in the center of each image, as there both, the lateral and vertical intensity gradients, are at their respective minimums (for geometric reasons)

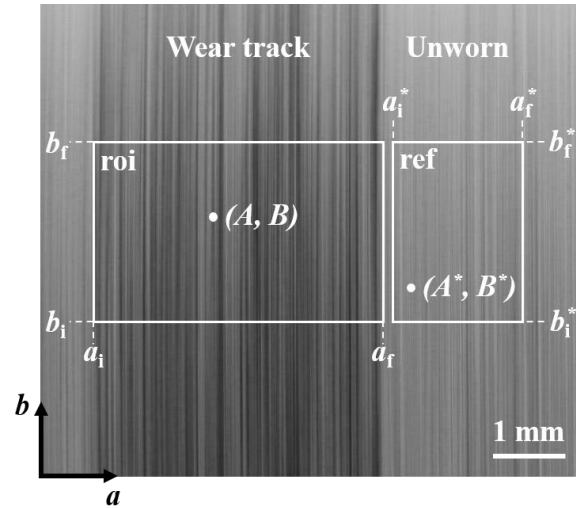


Figure 47: Exemplary definition of roi and ref in a circumferentially averaged photograph

Using these boundary conditions, the vertically averaged pixel value at horizontal position A can be defined as:

$$N^{\text{roi}}(A) = \frac{1}{\Delta B} \sum_{b=b_i}^{b_f} N(A, b) \quad (7)$$

$$N^{\text{ref}}(A^*) = \frac{1}{\Delta B^*} \sum_{b=b_i^*}^{b_f^*} N(A, b)$$

Using this, the vertically averaged relative luminance at horizontal position A can be defined as:

$$l^{\text{roi}}(A) = \frac{1}{K} \cdot \frac{f^2}{t_{\text{exp}} \cdot S} \cdot N^{\text{roi}}(A) \quad (8)$$

$$l^{\text{ref}}(A^*) = \frac{1}{K} \cdot \frac{f^2}{t_{\text{exp}} \cdot S} \cdot N^{\text{ref}}(A^*)$$

Physically, $N^{\text{roi}}(A)$ represents the vertically averaged luminance of a cylindrical section of the ring surface which is projected onto all pixels which share a common horizontal position A . As stated earlier, the fundamental principle of the presented

method is to measure the change of the luminance of the wear track due to the absorption of light by a transfer film that forms during the experiment. However, at this point the individual luminance still cannot be computed because equation (8) contains the camera's calibration constant which typically is unknown and not easily obtainable. Therefore, using the average reference area pixel value

$$N^{\text{ref}} = \frac{1}{K} \cdot \frac{f^2}{t_{\text{exp}} \cdot S} \cdot N^{\text{ref}}(A^*) \quad (9)$$

the average reference area luminance can be defined as

$$l^{\text{ref}} = \frac{1}{K} \cdot \frac{f^2}{t_{\text{exp}} \cdot S} \cdot N^{\text{ref}} \quad (10)$$

which in turn can be used to define the uncorrected¹ relative luminance at A as:

$$l_{\text{rel}} = \frac{100 \%}{l^{\text{ref}}} \cdot l^{\text{roi}}(A) = \frac{100 \%}{N^{\text{ref}}} \cdot N^{\text{roi}}(A) \quad (11)$$

Despite its simplicity, this equation has some very important implications: First, within an individual wear test, exposure time, ISO sensitivity and aperture number could theoretically vary wildly from image to image. While it is advised to keep them constant throughout an individual wear test, it still means that between tests or projects, lighting (and therefore exposure time, aperture number and ISO sensitivity) and camera model can be changed without affecting relative luminance. This property makes the presented method robust versus long-term changes in lighting, e.g. due to LED degradation and versus camera change, e.g. due to replacements for broken equipment. At the same time, it enables the methods use in different laboratories and the inter-laboratory exchange of relative luminance data.

Second, in order to compute the relative luminance at an in-roi pixel position A , only the vertically averaged pixel integer value for A , i.e. $N^{\text{roi}}(A)$, and the average pixel value of the reference area N^{ref} are required. Specifically, no corresponding pixel value $N^{\text{ref}}(a_1^* + A)$ is required. Therefore, the reference area does not have to be as wide as the roi. Specifically, it can be much smaller than the roi which is also the case in Figure 46. This is very handy with respect to the requirement of equivalent lighting conditions for roi and ref which is based on assumptions that get weaker the wider both areas get. Furthermore, much less unworn reference

¹ see section 5.3.5, page 66

area is needed which is very helpful when there are multiple wear tracks present on a test specimen.

Finally, yet importantly, substituting the noisy $N^{\text{ref}}(A^*)$ signal by an average also significantly improves the signal-to-noise ratio of the $l^{\text{ref}}(A)$ signal.

5.3.4 Data Processing, Aggregation and Plotting

Once both roi and ref are defined and applied to each image (by design – and conveniently – they are the same for all images of a given image series), the relative luminance is calculated at each A of every image. This yields a data set of relative luminance as a function of in-roi pixel position A and elapsed test time t . Then, by subtracting the relative luminance at the beginning of the experiment $l_{\text{rel}}(0)$ which should be 100 %, from all $l_{\text{rel}}(t)$, the change in relative luminance due to transfer film formation $\Delta l_{\text{rel}}(t)$ at roi-pixel position A and test time t is obtained:

$$\Delta l_{\text{rel}}(A, t) = l_{\text{rel}}(A, t) - l_{\text{rel}}(A, 0) \quad (12)$$

Using the wear track width w , A can be transformed into the lateral wear track position x :

$$x(A) = \frac{w}{\Delta A} \cdot A, \quad (13)$$

or in absolute image coordinates:

$$x(A) = \frac{w}{a_f - a_i} \cdot (a - a_i), \quad (14)$$

for all $a_i \leq a \leq a_f$. x is 0 at the left and w at the right border of the roi. Applying this transformation to the $\Delta l_{\text{rel}}(A, t)$ data yields Δl_{rel} as a function of lateral wear track position and elapsed test time, i.e. $\Delta l_{\text{rel}}(x, t)$.

Figure 41 shows a so-called ' xt -plot' which is one suggested way to represent x - and t -resolved data sets graphically. Data are taken from an ASTM G137 block on ring test on polyphenylene sulfide (PPS) that contains 40 wt.-% graphite. The test was done in dry sliding dry against an AISI 52000 steel ring, ground twist-free to $R_a = 0.14 - 0.19 \mu\text{m}$, $R_z = 1.5 - 1.9 \mu\text{m}$ and hardened to 60 – 62 HRc in standard laboratory atmosphere (21 – 25 °C, 40 – 60 % rH). In horizontal direction, the progression of test time is shown while in vertical direction the lateral wear track position is shown. Δl_{rel} is then represented by a grayscale color. According to the measurement principle, darker regions represent higher light absorption due to more

extensive transfer film formation. From this example, it can be seen that the formation of the transfer film is spatially and temporally inhomogeneous.

While x -resolved data sets contain the full information that can be extracted from a series of recorded images by the described procedure, a slightly higher level of data aggregation is reached by averaging the data across the whole wear track width:

$$N^{\text{roi/ref}}(t) = \frac{1}{\Delta A} \sum_{A=A_i}^{A_f} N^{\text{roi/ref}}(A, t) \quad (15)$$

$$l_{\text{rel}}(t) = \frac{1}{\Delta A} \sum_{A=A_i}^{A_f} l_{\text{rel}}(A, t) \quad (16)$$

$$\Delta l_{\text{rel}}(t) = l_{\text{rel}}(t) - l_{\text{rel}}(0) \quad (17)$$

The right side of Figure 42 shows exemplary plots of the laterally averaged data from the left side of the same Figure. At the top, the averaged pixel values of roi and ref are plotted as a function of elapsed test time. The constant, non-waivering N^{ref} -line indicates that the lighting conditions were very constant over the whole duration of the experiment. Actually, this is the prime purpose of this kind of data set and this kind of plot.

At the same time, from the N^{roi} -line a first overview can be gained concerning overall transfer film formation, the speed at which any transfer film has formed and whether a steady state in terms of luminance change due to transfer film formation has been reached during the testing time. If a steady state has been reached, its boundaries can be determined and used for computing the steady state average of $\Delta l_{\text{rel}}(t)$. As the simplest of solutions, start and end of the steady state can be determined by visual assessment. When t_i is the beginning and t_f is the end of the transfer film steady state, the steady state average of $\Delta l_{\text{rel}}(t)$ is computed according to:

$$\Delta l_{\text{rel,avg}} = \frac{1}{m} \sum_{t=t_i}^{t_f} \Delta l_{\text{rel}}(t) \quad (18)$$

Where m is the number of recorded images between t_i and t_f . In the shown example steady state has been identified to be between 10 and 20 h. Using this, a $\Delta l_{\text{rel,avg}}$ of -27% has been computed.

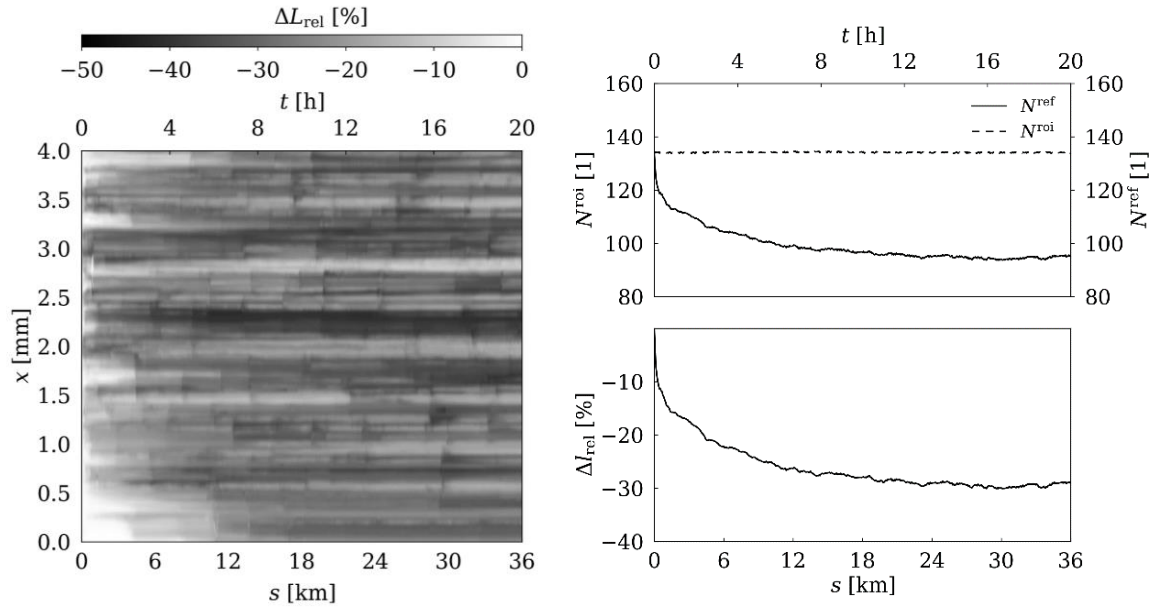


Figure 41: Exemplary plot of ΔL_{rel} as a function of lateral wear track position x and elapsed test time t (left) and Exemplary plot of $N^{\text{roi/ref}}(t)$ (top right) and of $\Delta L_{\text{rel}}(t)$ (right bottom).

5.3.5 Polishing Detection

Revisiting Figure 38, the elementary assumption of the presented method is that information about the state of the wear track surface in general and specifically on any formed or non-formed transfer film is transported to the camera sensor by light rays like ray \mathcal{F} . The light picks up this information by interacting with the ring surface and with any polymeric debris that is deposited there in form of a transfer film. Therefore, the idea is that detected luminance changes are a direct consequence of the formation of a transfer film. Specifically, when the luminance decreases, it is assumed that this is due to the partial absorption of light on the ring's surface by the transfer film.

However, there is (at least) one alternative process that also reduces the intensity of the light that reaches the camera after interacting with the ring. This happens when the ratio of specular and diffuse reflection shifts towards specular reflection. Then, the intensity of ray \mathcal{F} would be reduced significantly while specular reflection from point \mathcal{E} would increase. This happens predominantly when the wear track is polished, e.g. by the abrasive action of the polymer or of fillers included therein. Glass fibers and hard ceramic particles are examples for such fillers. Figure 50 shows an example for such a situation: of the four wear tracks, tracks 1 and 3 are

fully covered with a polymeric transfer film. Neither visual nor microscopic inspection reveal any degree of polishing. Wear track 2 is polished in its (lateral) center region while at its borders a transfer film has formed, see Figure 43. Note how polishing can locally increase or decrease brightness depending on the light's angle in incidence, camera position and ring curvature.



Figure 42: Example of local increase or decrease of brightness due to polishing, depending on angle, camera position and ring curvature.

For both shown wear tracks (1 and 2) the region of interest exhibits a lower luminance than their corresponding reference areas (not shown). For wear track 1, the sole reason for this is the formation of a light absorbing transfer film. For wear track 2 the formation of a transfer film also contributes to the observed loss of luminance. However, there is also a contribution from the polishing in the center of the track, where the steel has been polished.

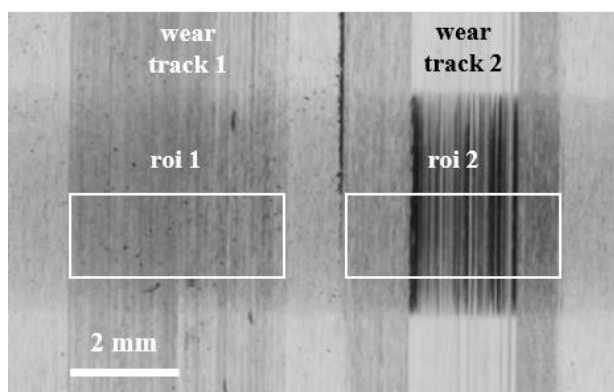


Figure 43: Post-test image of wear tracks 1 and 2: while track 1 lacks any polishing, track 2 is polished at its center.

Figure 44 shows the numeric $\Delta l_{\text{rel}}(t)$ data that is obtained by following the procedure described so far. As the image on the top is a post-test photograph of the wear tracks, it only corresponds to the last data point of the shown $\Delta l_{\text{rel}}(t)$ curves. To the bare eye, in the regions where there actually is a transfer film the change in luminance looks similar. However, when applied to the whole wear track width, the algorithm computes a much higher change in luminance for wear track 2 (−36 %) than for wear track 1 (−20 %).

The reason for this discrepancy is the significant contribution of the polished center area of wear track 2 to total luminance change.

The major issue with polishing is that it too can result in a luminance decrease. It is therefore not possible anymore to assume that any observed decrease in luminance is automatically the result of the formation of a transfer film. It might as well be the result of a polishing of the wear track; or a combination of both, see the given example. In order to discern between them solely on a numeric basis, i.e. on a basis suitable for an algorithm, additional information is required. This additional information can be obtained by considering ray traces of the lighting setup that so far have been ignored, see Figure 45 which is an expanded re-visitation of Figure 38. Due to the curvature of the ring, there are regions in each image in which the reflection of the light happens by a different combination of diffuse and specular reflection. When the steel ring is matte, the predominant reflection mechanism is diffuse reflection. This is the case when the ring has the correct initial surface preparation and as long as the ring is not polished. Because diffuse reflection, by definition, reflects light into all directions, some of the reflected light also reaches the camera, see the top left schema in Figure 45.

However, when the ring is polished in the wear track over the course of an experiment, specular reflection becomes the predominant reflection mechanism, see the top right schema. In this situation, virtually no light at all reaches the camera. This is because all light that originates from the inner walls of the light dome is reflected onto another point at the inner wall due to of the law of specular reflection and the convex curvature of the ring. The only light that hypothetically could be reflected from the roi to the camera would have to be emitted from the camera itself. From this, it is easily understood why polishing results in such a drastic decrease of the amount of light that reaches the camera from the roi.

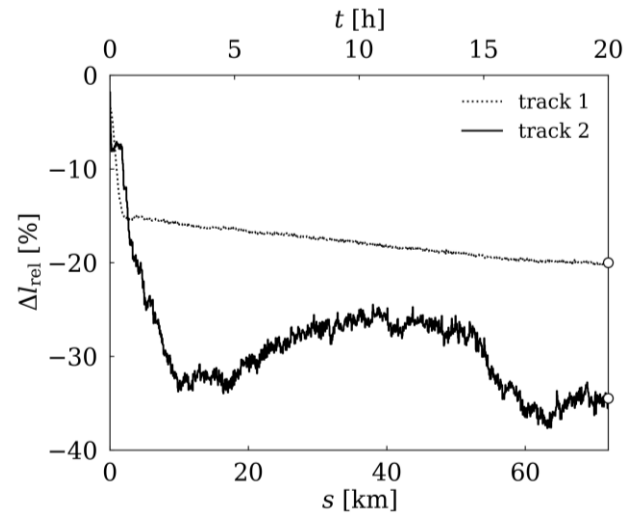


Figure 44: (Uncorrected) changes of relative luminance for wear tracks 1 and 2. The markers indicate the data points that correspond to the situation shown in the post-test image in Figure 43.

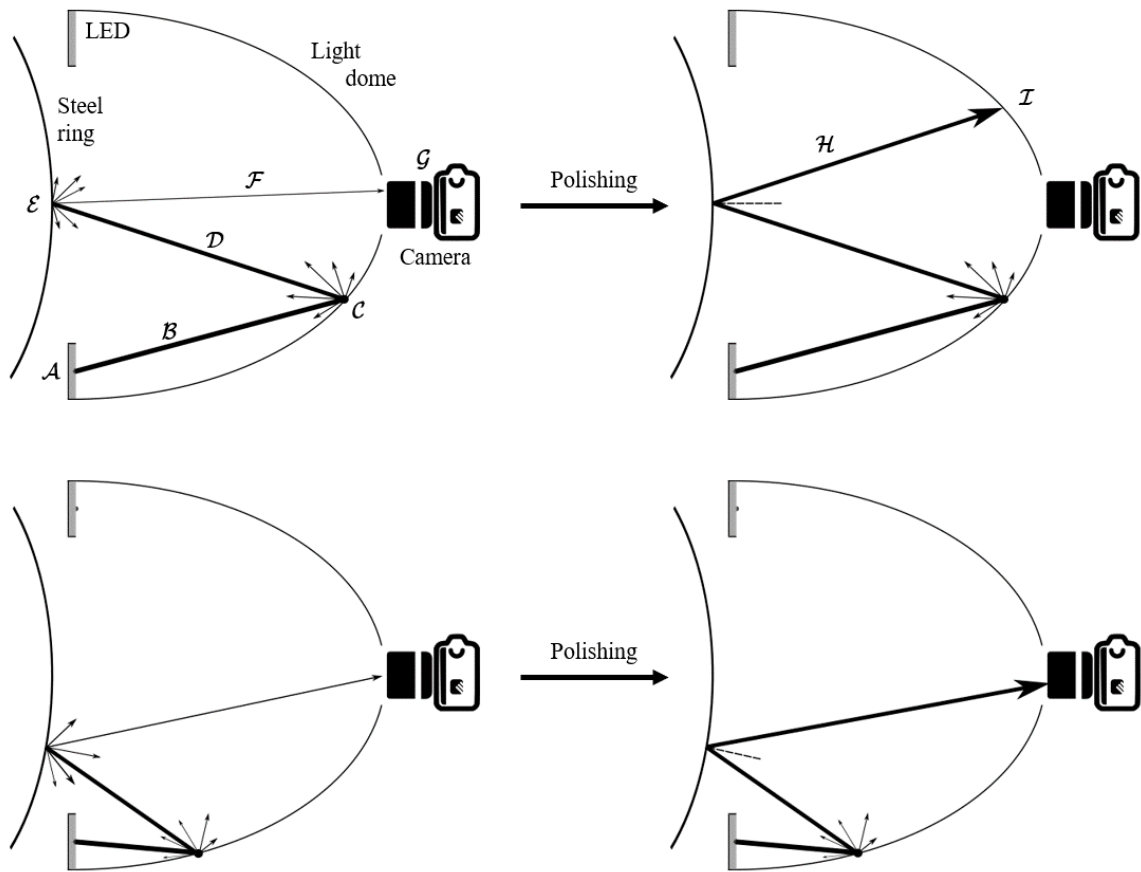


Figure 45: Effect of wear track polishing on ray traces. Top left: diffuse reflection in the center region of the image, some light reaches the camera because diffuse reflection emits light into all directions, including the direction of the camera. Top right: exclusive specular reflection in the center region of the image due to polishing, no light reaches the camera anymore. Bottom left: diffuse reflection below the center region of the image, some light reaches the camera. Bottom right: specular reflection due to polishing of the wear track – now much more light than with diffuse reflection reaches the camera.

Due to the curvature of the steel ring, the impact of polishing on the amount of light that reaches the camera significantly depends on the vertical position of the recorded image. Consider the two schemata at the bottom of Figure 45. When the ring is perfectly matte, the only light that reaches the camera from the area below (or above, for symmetry reasons) is light that is reflected diffusely. However, due to the special geometry of the test setup, outside of the center region of the image there is always a ray trace that satisfies the law of specular reflection with respect to the camera. Therefore, for real world surfaces, there would be a non-nil contribution of specular reflection to total reflection in these areas, and therefore they would look brighter than the central region, even for surfaces that look matte to the eye. This is in perfect agreement with the observation from Figure 43.

When the wear track is polished, the scenario from the bottom right schema of Figure 45 becomes effective: the polishing of the wear track results in an increase of the amount of light that reaches the camera from the non-center regions.

This is the direct opposite of the effect that polishing has on the roi in the center region of the image. The special importance of this is that while at the center region both transfer film formation and polishing result in a decrease of luminance, this is different in the non-center regions: while the formation of a transfer film results in a decrease of the intensity of the diffusely reflected light due to absorption, polishing results in an increase of the

intensity of the reflected light. This is the additional information which is needed to decide whether a decrease of the luminance in the roi is caused by either transfer film formation or by polishing.

In order to formalize this realization, we define two new image evaluation areas: the polishing detection region (pol) and the polishing detection reference region (pol-ref). Figure 46 shows the location of these two new evaluation regions. In an analogy to the restrictions for defining the reference area, see equation 6, the following applies to the definition of the pol- and the pol-ref area:

- The pol region must start and stop at the same horizontal pixel position than the roi region. Therefore, roi and pol are of the same width.
- The pol-ref region must start and stop at the same horizontal pixel position than the ref region. Therefore, ref and pol-ref are of the same width.
- pol and pol-ref must be of equal height.
- pol and roi should be in close vertical vicinity of each other.

Using definitions for the average pixel values of the pol region N^{pol} and of the pol-ref region $N^{\text{pol-ref}}$ that are analogous to equation 7, a test parameter for polishing detection P can be calculated as:

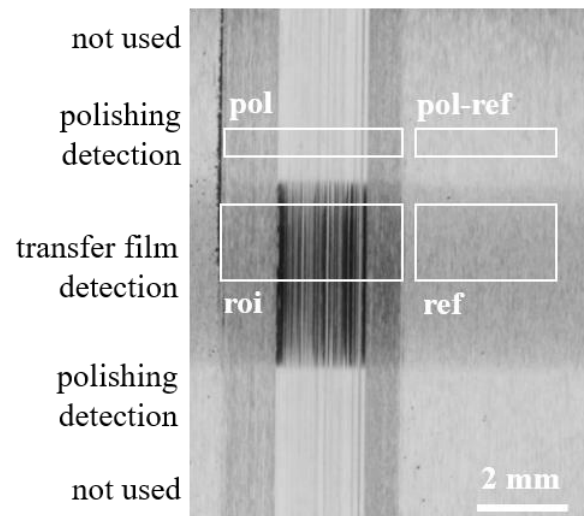


Figure 46: Exemplary definition of the evaluation regions needed for automated polishing detection. 1: roi, 2: ref, 3: pol, 4: pol-ref.

$$P(A, t) = \frac{N^{\text{pol}}(A, t)}{N^{\text{ref}}(A, t)} \cdot \frac{N^{\text{ref}}(t)}{N^{\text{pol-ref}}(t)} \cdot 100 \% \quad (19)$$

With this definition, the no observable effect level is 100 %. Using equation 19, the x, t -resolved polishing test parameter data has been calculated for the two wear tests that have resulted in the wear tracks shown in Figure 43. The plots of the thus obtained data sets, which effectively are x, t -plots of P are shown in Figure 47.

At $t = 0$, the mostly dark areas indicate that the polishing parameter is yet close to the no observed effect level across the whole wear track. For wear track 2, how-

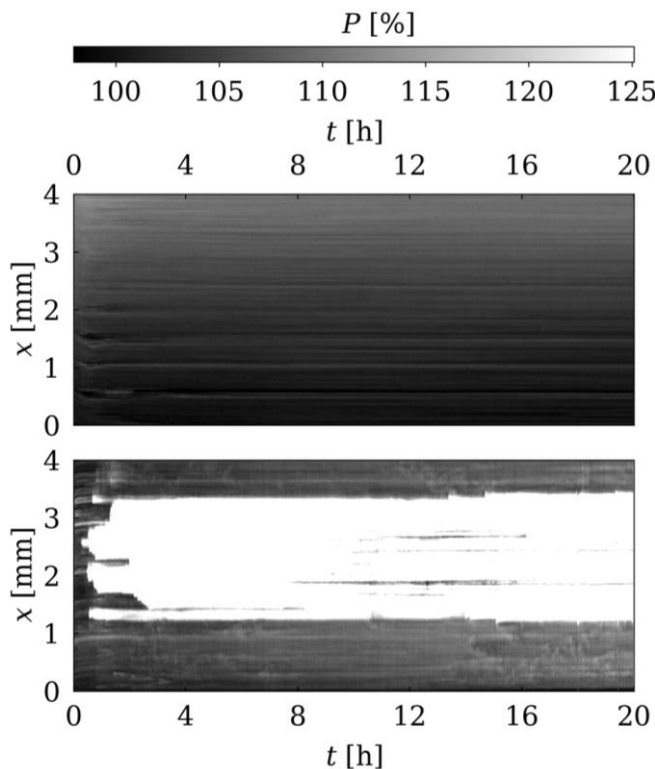


Figure 47: x, t -plots for exemplary wear tracks 1 (top) and 2 (bottom). The color scale's upper limit represents the utilized α value. Therefore, white color indicates the exclusion of the respective l data from the calculation of $\Delta L_{\text{rel}}(t)$.

ever, polishing sets on very soon as indicated by rising values of P .

As with transfer film formation, this happens at different elapsed times for different lateral wear track positions. For some intervals of x it does not happen at all during the test, e.g. from 0.0 mm to 1.1 mm and from 3.3 to 4.0 mm. The very right horizontal line of the graph directly corresponds to the situation shown in Figure 43. For wear track 1, the whole plot remains dark over the whole course of the experiment, which is consistent with the absence of any polishing as per the post-test photograph.

As P is a continuously distributed

variable, “unpolished” and “polished” must be discerned by a threshold α for P , above which l data must be considered to contain non-negligible contributions from polishing. 125 % has proven to tolerate a certain amount of (natural) scatter in the P data without giving false positives while still being sufficiently sensitive to actual polishing. However, this is only a suggestion for an otherwise arbitrary pick that the method's user must make.

Once the $P(x,t)$ information is fully extracted from the whole image series and a threshold is set, it can be used to exclude contributions to luminance changes from polishing. Therefore, any points in an $l_{\text{rel}}(x,t)$ data set for which the corresponding polishing parameter $P(x,t)$ exceeds α must be excluded from the calculation of $l_{\text{rel}}(t)$. Computationally, this is achieved using the Heaviside function Θ which assumes the value 0 for all negative arguments and 1 for all non-negative arguments. Using the difference between the computed polishing parameter and the polishing detection threshold, i.e. $P(x,t) - \alpha$, as argument for the Heaviside function, the relative luminance that has been initially defined by equation 16 can be redefined in a polishing-free variant (indicated by the capital version of letter L) as:

$$L_{\text{rel}}(t) = \frac{1}{n^*} \sum_{a=a_i}^{a_f} l_{\text{rel}}(a, t) \cdot \Theta(P(a, t) - \alpha) \quad (20)$$

$$\Delta L_{\text{rel}}(t) = L_{\text{rel}}(t) - L_{\text{rel}}(0) \quad (21)$$

with

$$n^* = n - \sum_{a=a_i}^{a_f} \Theta(P(a, t) - \alpha) \quad (22)$$

The right side of Figure 41 shows the plots of $\Delta L_{\text{rel}}(t)$ for the two wear tests that correspond to wear tracks 1 and 2 from Figure 43. Comparing them to the plots of the non-corrected data, i.e. $\Delta l_{\text{rel}}(t)$ (see Figure 44), no change at all has resulted for wear track 1 while for wear track 2 a completely different curve is obtained. Now the last data points of the curves correspond to $\Delta L_{\text{rel}}(20 \text{ h}) = -20.1 \%$ for track 1 and $\Delta L_{\text{rel}}(20 \text{ h}) = -19.8 \%$ for track 2 respectively which is in perfect agreement with the visual impression from the post-test photograph.

As a last useful quantity, the polishing ratio $R(t)$, i.e. the ratio of polished wear track area to total wear track area can be computed for an image recorded at elapsed test time t according to:

$$R(t) = \frac{n^*(t)}{n(t)} \quad (23)$$

The right side of Figure 48 shows the (t) -plots for both exemplary wear tracks which basically are higher level aggregations of the x,t -plots for P from Figure 47. They too are in very good agreement with Figure 43: the last data point of the R -curve, i.e. $R(t = 20 \text{ h})$, for track 2 is about 53 % which very well correlates with the ratio of polished wear track area as seen from the post-test photograph. Also, the

absence of any polishing from the image of wear track 1 corresponds well with the flat-line that has been computed for the corresponding $R(t)$ -data.

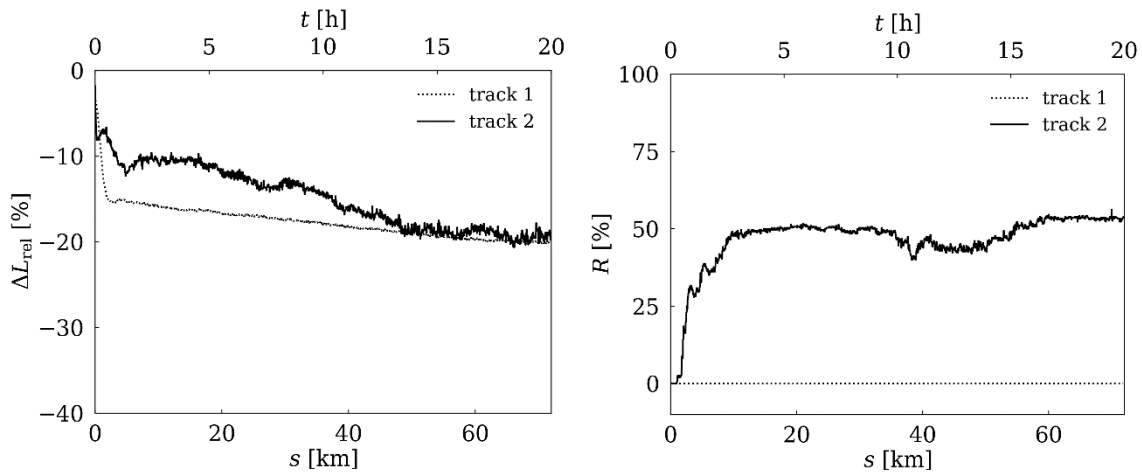


Figure 48: Polishing filtered version of the change of relative luminance (left) and polishing ratios for wear tracks 1 and 2.

5.3.6 Limitations and known issues

Based on the measurement principle and on the described evaluation process, the following limitations and issues exist:

Tarnishing: Starting at temperatures as low as 200 °C, steel surfaces are subject to thermally induced oxygenation which yields a thin and transparent iron oxide layer. Due to its transparency, it enables constructive interference of certain, height-specific wavelengths in the visible range which causes visible tarnish. Tarnish also changes the absolute luminance of the surface. Currently it is not clear, whether uniform tarnish will also affect relative luminance and should therefore be avoided until clarified. If tarnish is caused by frictional heat, there is a high chance that tarnish exhibits a lateral gradient. In this case, tarnish will falsify the calculation of relative luminance, as it affects region of interest and reference area differently. Therefore, when planning experiments for photo-optical luminance, test specimen and tribometer geometry with a good thermal conductivity should be preferred, oxidation resistant materials should be used and good cooling should be provided.

Thermal light emission: Significantly above the onset of tarnishing, starting at about 500 °C, metals start to emit visible light which will falsify the recorded absolute and relative luminance. More specifically, it will mix luminosity (the amount of light emitted by a body) with luminance (the amount of light reflected by a body) in a poten-

tially undefined way. While surely out of scope for most polymer/metal-slide pairings, this limitation should be minded when applying the method to high temperature sliding of more temperature resistant materials, e.g. ceramic/metal pairings.

Thermally induced changes of geometry: Thermal expansion and contraction due to changes in temperature also affect tribometer components. This is an issue if the observed test specimen, e.g. the ring, changes position within the recorded images. This can happen if the sample mount expands thermally when temperature changes significantly over the course of the experiment. There are three strategies to prevent this issue: compensation of thermal expansion by shifting the camera's position in sync with any thermal shifting of test specimen position, computational compensation of thermal shift and avoiding thermal expansion, e.g. by pre-heating the test setup or by not changing the mount's and specimen's temperature excessively (for IVW's *Atlas* Block on Ring tribometers 50 °C has proven to be a valid threshold below which thermal shifting is negligible).

Roughness: Due to what is described in section 5.3.5, photo-optical luminance analysis requires a minimum surface roughness. The exact value depends on the material and the geometry of the photographed test specimen, on the type of surface finish (ground twist-free, ground twisted, sandblasted, etc.) as well as potentially on the used lamps and camera equipment. In the case of IVW's *Atlas* Block-on-Ring tribometers, minimum R_a was about 0.10 μm .

Absorption: Photo-optical transfer film analysis is based on measuring the luminance of the test specimen on which the wear track is formed. Therefore, this test specimen should not absorb too much light as otherwise the signal-to-noise ratio might deteriorate. Currently, no threshold for the upper absorption limit can be specified; however, it will likely depend on the same factors than the minimum roughness limit.

Time resolved, but ex-post: Although yielding time-resolved data, transfer film luminance analysis is still an ex-post method, i.e. when the analysis is performed, the actual experiment is already over. Therefore, when an "event of interest", e.g. partial spontaneous transfer film degradation, is identified through the analysis, it is not possible anymore to investigate the involved specimen, e.g. with microscopy.

5.4 Automation and integration into *Atlas*

Temporal resolution of TLA data is achieved by taking images series. The combination of the inter-picture latency (“recording interval”) determines the temporal resolution of the resultant TLA data and the combination with the duration of the test determines the number of pictures that a TLA series consists of. With one image every ten seconds, typical 20 h tests yield 7,200 images. For a multi-position tribometer, this number is compounded by its number of positions. Additionally, a laboratory may operate multiple tribometer with TLA equipment. Therefore, the number of images that need to be evaluated over the course of one week may be in the range of tens to hundreds of thousands. Such amounts of data can only be evaluated automatically using a suitable script.

While such scripts can be designed to work outside of a LIMS (“stand-alone scripts”), the typical issues of unstructured and distributed data storage and processing would arise from this: data and results would be related to each other using naming conventions or convention-based locations on storage media. The same is then true for all kinds of derivative works from the original primary data, including processed, combined and derived data, result plots and reports. Furthermore, for proper operation, such scripts require input data. In the case of TLA, this would be the position of the four regions (roi, ref, pol and pol-ref). For this, these regions need to be defined which needs to be done only once for each series and which is therefore typically done manually. However, this sub-step requires the user to review a large number of individual images from different sections of the test and therefore of the corresponding image series. Here, manual selection and definition of the evaluation regions is virtually impossible, as with every newly assessed image, it must be decided, whether the previously chosen regions are still valid for the current image. This process is then iterated for several sections of the image series until the user has decided that the current selection of regions is a good enough fit for the whole image series. If, at any point he or she decides that a given region definition only adequately matches a subsection of the image series, the selection needs to be adjusted and the whole process needs to be restarted (i.e. any previous partial assessments become invalid when the region definition is modified). This makes the manual review of an image series virtually impossible without the aid of a specialized software tool for region definition. Still, even when

regions have been determined, they need to be transferred to the evaluation script. Additionally, any evaluation script will need the width of the test specimen which is used as an estimate for the physical width of the investigated wear track and which is therefore used to calibrate pixel sizes to physical dimensions.

When the evaluation has been done, the obtained numerical results need to be joined with other sensor data, e.g. with friction forces or test specimen height loss.

Due to the large amount of data that is obtained from TLA evaluation, this can only be done programmatically, requires a high level of user training and experience in terms of programming and data handling. Given the combination of task difficulty, human error rates for even basic tasks and the breaking of integrity of a large amount of data with even small errors, the manual execution of this sub-step prone to high error rates.

Overall, the evaluation of TLA data cannot be safely done without being linked to or integrated into a LIMS and without the establishment of automated retrieval and transmission of required input data and automated handling and processing of evaluation results. For *Atlas*, a fully integrated process for the evaluation of TLA image series has been implemented for which Figure 49 shows the workflow. This ensures the usage of the correct evaluation script at all times, enables the use of the parallel computing capabilities of the *Atlas* server and alleviates the user of any responsibilities for data handling, processing and storage. Therefore, this also reduces the

requirements for user training and the occasions of possible human errors to a minimum. The first sub-step, the upload of the image series to *Atlas*, can be achieved with a process that is analogue to the upload of measurement data files,

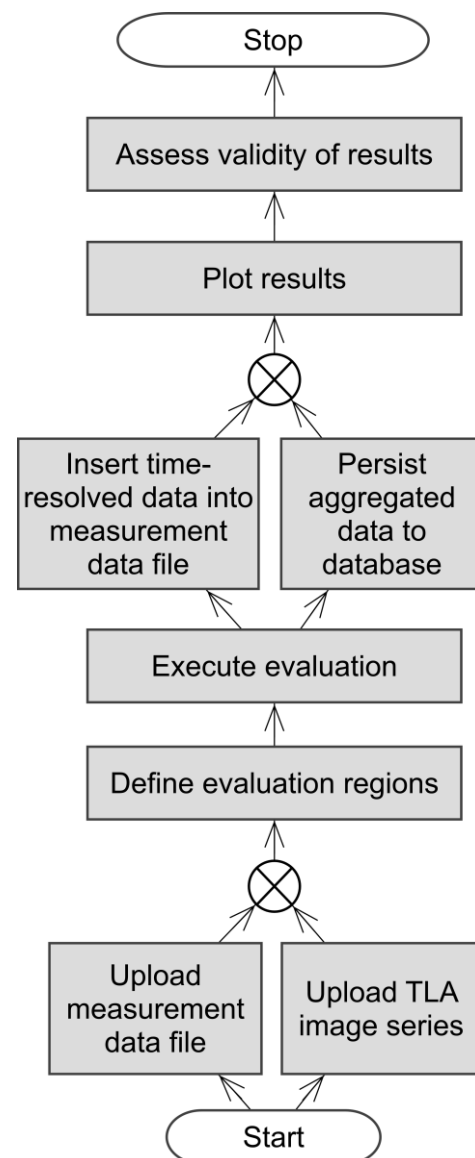


Figure 49: Workflow of TLA evaluation with *Atlas*. Grey: computerized support, white: manual.

see Section 4.9.6. *Atlas* will then extract the image series from the uploaded zip file and transfer the images to the storage location of the corresponding wear test.

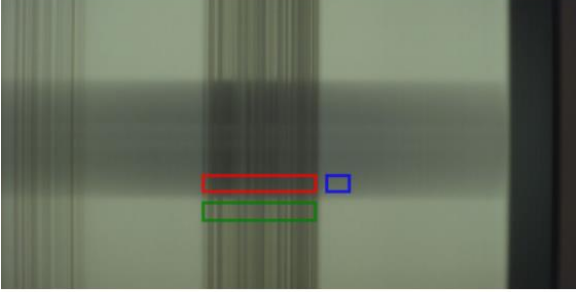
Next, the user needs to define the four regions required for TLA evaluation. As pol-ref is fully determined by the position of ref and pol, the user can actually only determine roi, ref and pol. Figure 50 shows *Atlas*' tool for this.

It is written with JavaScript and executed in the user's browser. By shifting the position of the slider below the image, images from different sections of the wear test can be displayed (which are loaded from the server dynamically). Then, by using the three mouse buttons, the user can draw three regions. By holding the left mouse button clicked, the roi can be drawn in red. Correspondingly, holding the right or middle mouse button lets the user draw

the ref or the pol region. While drawing the different regions, the tool will automatically enforce the business rules that apply to the boundaries of these regions, see Sections 5.3.3 and 5.3.5, thereby greatly reducing the risk for human error of the "failure to apply business rules"-type. When shifting the slider to different sections of the test, the defined regions are persisted on the canvas. This greatly simplifies the assessment process described above and furthermore eliminates sources of human error (applying wrong regions to images, accepting a sub-standard region definition for reduced effort). The tool then provides convenience functions for capturing the coordinates of the defined regions and for submitting the definition to the server and start the evaluation process. Again, this reduces the possibility of human error that would be associated with a hypothetical manual process (error rates [96]: read 10-digit number wrongly: 0.6 %, type character wrongly: 1.0 %).

When *Atlas* receives the information on the boundaries of the evaluation regions, it will execute the following workflow:

Luminosity image series for wear test 550184



	in-track	reference	polishing
x-start	504	812	504
x-stop	784	868	784
y-start	661	661	729
y-stop	701	701	773

Buttons: Capture coordinates, Download image, Submit and start upload, Rotate 90

Figure 50: Software tool provided by *Atlas* for determining TLA evaluation regions ("segmentation") and for starting the evaluation.

Load existing test data: The results of the overall processing of the image series ultimately needs to be persisted to a test data file. Therefore, the measurement data file is loaded. If it does not exist, the processing of the image series is aborted.

Image series filename normalization: In order to accelerate image processing, the file name format of the image series is normalized. The default file name for any given individual image is based on the time at which it was recorded. This information is extracted either from the metadata stored in the image (“EXIF”, Exchangeable Image File) or from its current file name (needs to be a timestamp).

Synchronization: Either from the database or from the loaded test data file, a timestamp for the start of the wear test is obtained. Also, a timestamp for the recording time of the first image of the image series is obtained. Then the time difference between the two is computed and the timestamps of all images of the image series are then tared. After this, any potential desynchronizations between recorded sensor data and the image recording software are eliminated. If necessary, the images’ EXIF data and file names are update to reflect the changes.

Compile image list, set up local variables: As synchronization potentially changes the names of the images on the server-side storage, a new list of image file names is compiled. Before the list of image file names is iterated, a set of local variables is computed which contains all information that is needed for processing the individual images and for storing the results of the performed calculation. As they do not change in-between images, they are computed once before iteration is started.

Process images: The images are processed by iterating over the list of image file names that was created previously. Single images are then processed using ImageMagick [97]. Although this is done in a separate operating system process, the corresponding Ruby call for this is synchronous, i.e. it waits for the end of the execution of ImageMagick. Therefore, a simple iteration over the image series would result in sequential processing of the individual image. In order to enable concurrent processing, the calls for processing individual images must be parallelized, e.g. using threads. As CRuby is not inherently thread-safe, or more specifically, the concurrent access to its compound data types like arrays and hashes by threads of the same process is not thread-safe, the actual iteration over the data structure that holds references to the individual images of a series must be made thread-safe using Ruby’s `Mutex` implementation, see Code listing 29.

Code listing 29: Concurrent processing of images using mutual thread exclusion.

```

# initialize concurrency
threads = []
mutex = Mutex.new
# no of threads is dynamically computed from actual hardware configuration
no_of_threads.times{ |j|
  threads << Thread.new do
    loop do
      timestamp = nil; image = nil
      mutex.synchronize{ timestamp, image = image_list.shift }
      break if (image.blank? or timestamp.blank?)
      # process image by system call to ImageMagick
    end
  end
end
end
end

```

When called, ImageMagick will convert the corresponding image to greyscale and then segment it into the four pre-defined evaluation regions. In each region, the pixel values are averaged in vertical direction as per equation (7). Data is then further averaged in horizontal direction in order to obtain scalar pixel values for each region according to equation (15). Using the x -resolved pixel values and the region averages, polishing detection signals according to equation (19) are computed for each lateral pixel position of the region of interest. Using these, uncorrected and corrected relative luminance according to equations (16) and (20) are computed in x -resolved manner as well as an arithmetic average for the whole image. Also, the polishing ratio R is computed according to equation (23).

Merge data: After all images are processed, time-resolved scalar results of the image analysis are integrated into the measurement data file. These are x -averaged pixel values of roi and ref as well as the x -averaged versions of corrected and uncorrected luminance and the polishing ratio. As each segment of a wear test has its own data file, luminance analysis results are distributed across these files based on timestamps. Now, these data are part of the test's data file(s) and can therefore be included in all subsequent evaluation operations.

Save data: Finally, the merged data files are saved to disk. Also, the x,t -resolved data, which do not fit into the standard wear test data files (as they are no single scalars as a function of elapsed time) are written to separate data files which contain x,t -resolved data of N^{roi} , N^{rel} , L_{rel} , L_{rel} , ΔL_{rel} and P and which thus have "xt" as file extension. One file is created per wear test segment. Such files are plotted by instances of XtPlotGraph, see the left side of Figure 41 for an example.

6 Summary

In tribology laboratories, a wide range of processes is carried out in order to investigate friction and wear. These include critical operations like identifying and storing test specimens, defining individual wear tests and aggregating them into test series and projects, parameterizing tribometers before test conduction, the transmission, evaluation and storage of recorded data files, test setup and test specimen photographs as well as of micrographs and profilometric data. Additionally, all of these processes yield large amounts of metadata, e.g. in the form of timestamps (when was a test done?) or relationships (on which tribometer was a test done?). Until today, these processes are mainly conducted manually. Some of them, like tribometer configuration and data handling have however become highly complex, which gives rise to significant error rates.

While other fields of science, e.g. clinical medicine or biomedical research have long-since developed Laboratory Information Management Systems for supporting their operations, to reduce human error and to improve data quality, no such system currently exists for tribology laboratories beyond some narrow-focused proprietary solutions. This work therefore deals with the design and implementation of *Atlas*, a LIMS that is specifically designed for tribology laboratories. For this, an analysis of a typical workflow for such a laboratory yielded a collection of physical entities and processes that are crucial for the main workflow of tribology laboratories: test specimens, tribometers, wear tests, tribometer parameterization, measurement data files, the evaluation of wear tests and the visualization and classification of test results. For the physical entities, data models have been formulated using object-relational data modeling. This included the formulation their respective attributes (including their data types) and the implementation of their respective domain-specific behavior using object-oriented programming.

For test specimens, a novel three-tiered, hierarchical data model has been developed, which consists of test specimen classes, lots and of individual test specimens. Its three layers closely reflect the different levels on which information is collected and stored either on the level of individual test specimens (like customer specific labels), on the level of a specimen lot (like surface roughness of rings after machining) or on the most abstract level, the test specimen class level (like the material that test specimens are made of). Atypical inverse property propagation

has been developed to reflect the business rule of subsidiarity: data from the level of individual test specimens supersedes data stored on the two higher levels, but if no individual data is available, higher-level data should replace the missing data on the fly. This gives users the ability to define the granularity of their data collection on a by-case basis, thereby contributing greatly to reducing effort when individual data collection is deemed unnecessary.

For tribometers, a compound data model has been developed, that consists of the actual tribometer and a one-to-many relationship to testing positions. This enables the representation of multi-position tribometers while it still includes the single-position design as edge cases. Specifically, this data model enables a semantic relationship between wear tests and tribometers, that not only includes the exact mounting position of a tests' specimens in a multi-station tribometer but also allows for the inclusion of additional information, like geometry-related conversion factors from raw sensor data to corrected physical quantities, into this relationship.

For wear tests, a data model has been developed that reflects the many different types of wear tests that are done even in fundamental research. Specifically, the established OOP concepts of inheritance, specialization and single table inheritance have been used to reproduce this variability of the domain in the data models and for the implementation of type-specifics, e.g. the use of different types of test specimens for different types of wear tests. Furthermore, the data model reflects the requirement that wear tests may be done as sequences of individual wear test segments, which are test sections with stationary sets of test parameters. The data model for wear test segments also uses OOP techniques to define segment subtypes for specific testing modes, like unidirectional sliding, oscillation, positioning or measurement of the static coefficient of friction. Combined with the different test subtypes, this provides the ability to create digital representations of virtually every test that is done today in a tribological laboratory.

While machine-to-machine communication is an established technology, its use in tribology for the parametrization of tribometers is so far limited to a few proprietary solutions. However, this process is highly prone to error when done manually. Here, the machine-readable storage of all relevant test parameters inside *Atlas* has proven to be the perfect basis for the generation of machine-readable documents that describe the parameterization that is required to execute a given test. Depend-

ing on support by the tribometer control software, three different workflows are presented: test parameterization data can either be pushed from *Atlas* to the tribometer or pulled from the tribometer software via an HTTP request that includes the test(s) IDs. Additionally, for the case of network failures or tribometers that are operated offline, parameterization data can be downloaded from *Atlas* through a web browser. The use of either of these procedures eliminates the high rate of error that is typically observed for manual parameterization of tribometers. At IVW's tribology laboratory, this has been so successful that manual parameterization has been disabled on most its tribometers.

For measurement data files, a standardized data format has been defined. However, tribometers typically produce a wide variety of data format. Ruby's singleton methods have been presented as a viable solution for transforming non-standard compliant data files in standard compliant versions. Using life cycle hooks defined by Ruby on Rails, these are dynamically generated from database-persisted code via metaprogramming. Based on the assertion of only storing standardized data files, which include machine-readable column descriptors, the procedure on how to implement an automated evaluation of tests in general and specifically of data files is presented. Here, the reduction of complexity while still maintaining a high degree of specialization by the use of the OOP concepts of inheritance, duck typing and (partial) method overriding have been presented.

Another benefit of standardized data files that has been presented, is the possibility of automating the visualization of measured data series, like COF versus sliding distance ("recorded data graphs"), and of aggregated data like average COF in steady state versus sliding speed ("aggregate data graphs"). Here, *Atlas* again uses object inheritance to produce a hierarchical data model whose individual components reflect different types of graphs that are typically used in tribology.

As this thesis does not simply provide ready-made solutions but describes the rationale and tools of modeling wear tests, in combination with its exclusive use of open and free technologies, it lays the foundation for the extension of the sample implementation in order to enable users to implement missing features and meet demands that arise from future developments in the field of tribological testing. It demonstrates this by the example of Transfer Film Luminance Analysis (TLA) which is a novel method for detecting and quantifying the formation of transfer films and their impact on friction and wear and which has been newly invented within

the scope of this work. Transfer films form by the transfer and adhesion of wear debris during virtually all types of tribological contacts and operation modes, mainly however in sliding and rolling. For dry sliding pairings, they critically define friction and wear. However, until today, no method exists for investigating the build-up and the stability of such films and their impact of friction and wear that is universally applicable, laterally extensive, quantitative, in-situ and time-resolved. While TLA, solves all of these problems, its evaluation procedure is highly complex. Specifically, it requires the retrieval and transmission of test parameters and other input data, specific programming techniques for concurrent image processing and it yields a large amount of data that needs to be handled. Here, the implementation of an automated evaluation in *Atlas* has resolved all these issues and has made TLA easy to use. Actually, while TLA and its automated evaluation are presented in a closed manner in this thesis, its development was tightly coupled with its implementation in *Atlas*.

Overall, using the techniques used in this work and the implementation of *Atlas* as an exemplary LIMS that is highly specific to tribology laboratories, many processes that have so far been executed manually can now be fully replaced by automated processes (like the transmission, standardization and storage of measurement data files) or at least by supported by highly-specialized software tools (like the generation of test specimen identifiers. As a result, these processes significantly become less error-prone and faster, which – in turn – greatly improves their efficiency. This frees up researchers' time by reducing time spent on repetitive, lowly standardized, complicated, demanding and therefore error-prone operations. Specifically, this has enabled the development of TLA, which is one of the few methods that can produce time resolved and quantitative data on the important topic of transfer film formation and that is the only method that, in doing so, is not subject to severe restrictions on the selection of materials, testing conditions and that does not require expensive testing equipment and extensive personnel training.

References

- [1] J. L. Sepulveda and D. S. Young, "The ideal laboratory information system," *Arch. Pathol. Lab. Med.*, vol. 137, no. 8, pp. 1129–1140, 2013, doi: 10.5858/arpa.2012-0362-RA.
- [2] P. J. Prasad and G. L. Bodhe, "Trends in laboratory information management system," *Chemometrics and Intelligent Laboratory Systems*, vol. 118, pp. 187–192, Aug. 2012, doi: 10.1016/j.chemolab.2012.07.001.
- [3] G. Everest, "BASIC DATA STRUCTURE MODELS EXPLAINED WITH A COMMON EXAMPLE.," Oct. 1976.
- [4] E. Santner, "Computer Support in Tribology - Experiments and Database," *Tribotest journal*, vol. 2, no. 267, pp. 267–280, 1996.
- [5] A. W. Ruff, "Friction and wear data bank," in *Modern Tribology Handb.: Volume One: Principles of Tribology*, CRC Press, 2000, pp. 523–561.
- [6] J. Rumble and L. Sibley, "Towards a tribology information system: the results of a planning workshop held at the National Bureau of Standards, July-August 1985.," *NBS Special Publication*, vol. 737, 1987.
- [7] S. M. Zakharov, "Computer tribology," *Trenie i Iznos*, no. 1, pp. 98–106, 1993.
- [8] C. D. National Institute of Standards and Technology (NIST), "NIST standard reference data products catalog 1994," *govinfo.gov*, Jan. 01, 1994.
- [9] Woydt, "Tribocollect - Tribological Database With More Than 10,000 Data Sets."
- [10] Y.-B. Xie, "On the tribology design," *Tribology International*, vol. 32, no. 7, pp. 351–358, Jul. 1999, doi: 10.1016/S0301-679X(99)00059-6.
- [11] M. Sedlaček, B. Podgornik, and J. Vižintin, "Tribological properties of DLC coatings and comparison with test results: Development of a database," *Materials Characterization*, vol. 59, no. 2, pp. 151–161, Feb. 2008, doi: 10.1016/j.matchar.2006.12.008.
- [12] A. A.-K. H. Abu-Ein, S. M. Fayyad, M. Al-Rashdan, and G. Al-Marahle, "Data-Base System of Tribological Coating Materials," *Contemporary Engineering Sciences*, vol. 5, no. 7, pp. 315–324.
- [13] F. Pirker, U. Cihak-Bayr, A. Vernes, and I. Tóth, "i-TRIBOMAT – the Open Innovation Test Bed for Tribological Materials Characterisation," presented at the 60. Tribologie-Fachtagung, Göttingen, Germany, 25.09 2019.

- [14] U. Cihak-Bayr, L. Katona, I. Tóth, and F. Birker, “LAB-2-Field: Up-scaling of tribological materials behaviour to real component behavior,” presented at the 60. Tribologie-Fachtagung, Göttingen, Sep. 2019.
- [15] D. Spaltmann, M. Kogia, S. Liedtke, D. Dykeman, and T. Gradt, “Challenges for the design of a universal tribological database for materials,” 2019, p. 37/1-37/5. Accessed: Dec. 04, 2021. [Online]. Available: <https://opus4.kobv.de/opus4-bam/frontdoor/index/index/docId/49147>
- [16] “Intelligent Open Test Bed for Materials Tribological Characterisation Services | i-TRIBOMAT Project | Results | H2020 | CORDIS | European Commission.” <https://cordis.europa.eu/project/id/814494/results/de> (accessed Dec. 04, 2021).
- [17] Z. Zhang, N. Yin, S. Chen, and C. Liu, “Tribo-informatics: Concept, architecture, and case study,” *Friction*, vol. 9, no. 3, pp. 642–655, Jun. 2021, doi: 10.1007/s40544-020-0457-3.
- [18] P. Kügler, M. Marian, B. Schleich, S. Tremmel, and S. Wartzack, “tribAln—Towards an Explicit Specification of Shared Tribological Understanding,” *Applied Sciences*, vol. 10, no. 13, Art. no. 13, Jan. 2020, doi: 10.3390/app10134421.
- [19] D. I. Inc, “MOOHA - Digital Laboratory Assistant.” <https://www.ducom.com/digital> (accessed Dec. 09, 2021).
- [20] S. Kuiry, “TriboScript Software Simplifies Tribological and Mechanical Test Procedures,” Bruker Nano Surfaces Division, San Jose, California, USA, Technical Note TN1007 Rev. A1, 2016.
- [21] P. P.-S. Chen, “The entity-relationship model—toward a unified view of data,” *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, Mar. 1976, doi: 10.1145/320434.320440.
- [22] P. P.-S. Chen, “English sentence structure and entity-relationship diagrams,” *Information Sciences*, vol. 29, no. 2, pp. 127–149, May 1983, doi: 10.1016/0020-0255(83)90014-2.
- [23] P. P.-S. Chen, “English, Chinese and ER diagrams,” *Data & Knowledge Engineering*, vol. 23, no. 1, pp. 5–16, Jun. 1997, doi: 10.1016/S0169-023X(97)00017-7.
- [24] C. W. Bachman, “Data structure diagrams,” *SIGMIS Database*, vol. 1, no. 2, pp. 4–10, Jul. 1969, doi: 10.1145/1017466.1017467.

-
- [25] R. Barker, *Barker, R: Case Method: Entity Relationship Modelling*, 1st ed. Harlow: Pearson Education, 1990.
- [26] J. Martin, *Information engineering, planning & analysis: book 2*. USA: Prentice-Hall, Inc., 1990.
- [27] J. Martin, "Information engineering : Book III design & construction," 1990, Accessed: Dec. 06, 2021. [Online]. Available: <https://library.ui.ac.id/detail?id=20309782>
- [28] "About the Unified Modeling Language Specification Version 2.5.1." <https://www.omg.org/spec/UML> (accessed Dec. 06, 2021).
- [29] G. B. / J. R. / I. J. B. / R. / Jacobson, *The Unified Modeling Language User Guide*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley Professional, 2005.
- [30] A. P. Black, "Object-oriented programming: Some history, and challenges for the next fifty years," *Information and Computation*, vol. 231, pp. 3–20, Oct. 2013, doi: 10.1016/j.ic.2013.08.002.
- [31] R. Elmasri and S. B. Navathe, *Fundamentals of database systems (2nd ed.)*. USA: Benjamin-Cummings Publishing Co., Inc., 1994.
- [32] M. Egenhofer and A. Frank, "Object-oriented modeling in GIS: inheritance and propagation," in *Proceedings of the International Symposium on Computer-Assisted Cartography*, Baltimore, Maryland, USA, 07.02 1989, pp. 588–598.
- [33] N. Milojkovic, M. Ghafari, and O. Nierstrasz, "It's Duck (Typing) Season!," in *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, May 2017, pp. 312–315. doi: 10.1109/ICPC.2017.10.
- [34] B. Meyer, *Object-Oriented Software Construction*, 2nd Edition. Prentice Hall, 1998.
- [35] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970, doi: 10.1145/362384.362685.
- [36] D. D. Chamberlin and R. F. Boyce, "SEQUEL: A structured English query language," in *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, Ann Arbor, Michigan, May 1974, pp. 249–264. doi: 10.1145/800296.811515.
- [37] "Metadata Definition & Meaning - Merriam-Webster." <https://www.merriam-webster.com/dictionary/metadata> (accessed Apr. 20, 2022).

- [38] S. Hitchman, "The details of conceptual modelling notations are important—a comparison of relationship normative language," *Communications of the Association for Information Systems*, vol. 9, no. 1, p. 10, 2002.
- [39] D. Colley, C. Stanier, and M. Asaduzzaman, "Investigating the Effects of Object-Relational Impedance Mismatch on the Efficiency of Object-Relational Mapping Frameworks," *JDM*, vol. 31, no. 4, pp. 1–23, Oct. 2020, doi: 10.4018/JDM.2020100101.
- [40] C. Ireland and D. Bowers, "Exposing the myth: object-relational impedance mismatch is a wicked problem," in *DBKDA 2015, The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications*, Rome, Italy, May 2015, pp. 21–26.
- [41] C. Ireland, D. Bowers, M. Newton, and K. Waugh, "A Classification of Object-Relational Impedance Mismatch," in *2009 First International Conference on Advances in Databases, Knowledge, and Data Applications*, Mar. 2009, pp. 36–43. doi: 10.1109/DBKDA.2009.11.
- [42] M. Fowler, *Patterns of Enterprise Application Architecture*, 01 ed. Boston: Addison Wesley, 2002.
- [43] P. Ritchie, "The security risks of AJAX/web 2.0 applications," *Network Security*, vol. 2007, pp. 4–8, Mar. 2007, doi: 10.1016/S1353-4858(07)70025-9.
- [44] "XMLHttpRequest." [https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms759148\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms759148(v=vs.85)) (accessed Dec. 04, 2021).
- [45] Gesellschaft für Tribologie e.V., "Arbeitsblatt 7 TRIBOLOGIE. Verschleiß, Reibung DEFINITIONEN, BEGRIFFE, PRÜFUNG," Gesellschaft für Tribologie e.V.
- [46] R. E. Kaiser and G. Gottschalk, *Elementare Tests zur Beurteilung von Meßdaten: Soforthilfe für statistische Tests mit wenigen Meßdaten*. Mannheim: Bibliographisches Institut, 1972.
- [47] "Standard Guide for Statistical Procedures to Use in Developing and Applying Test Methods." <https://www.astm.org/e1488-12r18.html> (accessed Dec. 08, 2021).
- [48] "The Ruby License." <https://www.ruby-lang.org/en/about/license.txt> (accessed Dec. 05, 2021).
- [49] "PostgreSQL: License." <https://www.postgresql.org/about/licence/> (accessed Dec. 05, 2021).

-
- [50] “The PostgreSQL Licence (PostgreSQL) | Open Source Initiative.” <https://opensource.org/licenses/postgresql> (accessed Dec. 05, 2021).
- [51] “GRANT,” *PostgreSQL Documentation*, Aug. 12, 2021. <https://www.postgresql.org/docs/12/sql-grant.html> (accessed Dec. 11, 2021).
- [52] D. Cherry, *Securing SQL Server: Protecting Your Database from Attackers*, 3rd ed. Waltham, MA: Syngress, 2015.
- [53] “ASTM G137 Standard Test Method for Ranking Resistance of Plastic Materials to Sliding Wear Using a Block-on-Ring Configuration.”
- [54] P. Bailis, A. Fekete, M. J. Franklin, A. Ghodsi, J. M. Hellerstein, and I. Stoica, “Feral Concurrency Control: An Empirical Investigation of Modern Application Integrity,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, May 2015, pp. 1327–1342. doi: 10.1145/2723372.2737784.
- [55] J. C. Wincek and J. M. Haight, “Realistic human error rates for process hazard analyses,” *Process Safety Progress*, vol. 26, no. 2, pp. 95–100, 2007, doi: 10.1002/prs.10184.
- [56] “Extensible Markup Language (XML) 1.1 (Second Edition).” <https://www.w3.org/TR/xml11/> (accessed Dec. 05, 2021).
- [57] “The JavaScript Object Notation (JSON) Data Interchange Format.” <https://datatracker.ietf.org/doc/html/rfc8259> (accessed Dec. 05, 2021).
- [58] “The Official YAML Web Site.” <https://yaml.org/> (accessed Dec. 05, 2021).
- [59] “Module: YAML (Ruby 3.0.1).” <https://ruby-doc.org/stdlib-3.0.1/libdoc/yaml/rdoc/YAML.html> (accessed Dec. 05, 2021).
- [60] “olbrich/ruby-units: A unit handling library for ruby.” <https://github.com/olbrich/ruby-units> (accessed Dec. 13, 2021).
- [61] E. Padenko, L. van Rooyen, B. Wetzel, and J. Karger-Kocsis, “‘Ultralow’ sliding wear polytetrafluoro ethylene nanocomposites with functionalized graphene,” *Journal of Reinforced Plastics and Composites*, vol. 35, no. 11, pp. 892–901, Jun. 2016, doi: 10.1177/0731684416630817.
- [62] G. Zhang, W. Österle, B. Jim, I. Häusler, R. Hesse, and B. Wetzel, “The role of surface topography in the evolving microstructure and functionality of tribofilms of an epoxy-based nanocomposite,” *Wear*, vol. 364–365, pp. 48–56, Oct. 2016, doi: 10.1016/j.wear.2016.06.012.

- [63] K. R. Makinson and D. Tabor, "The friction and transfer of polytetrafluoroethylene," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 281, no. 1384, pp. 49–61, 1964.
- [64] J. K. Lancaster, "Relationships between the Wear of Polymers and their Mechanical Properties," *Proceedings of The Institution of Mechanical Engineers*, vol. 183, no. 16, pp. 98–106, Sep. 1968, doi: 10.1243/PIME_CONF_1968_183_283_02.
- [65] J. K. Lancaster, "Polymer-based bearing materials: The role of fillers and fibre reinforcement," *Tribology*, vol. 5, no. 6, pp. 249–255, Dec. 1972, doi: 10.1016/0041-2678(72)90103-0.
- [66] S. H. Rhee and K. C. Ludema, "Mechanisms of formation of polymeric transfer films," *Wear*, vol. 46, no. 1, pp. 231–240, Jan. 1978, doi: 10.1016/0043-1648(78)90124-2.
- [67] S. K. Rhee, M. G. Jacko, and P. H. S. Tsang, "The role of friction film in friction, wear and noise of automotive brakes," *Wear*, vol. 146, no. 1, pp. 89–97, May 1991, doi: 10.1016/0043-1648(91)90226-K.
- [68] G. Zhang, I. Häusler, W. Österle, B. Wetzel, and B. Jim, "Formation and function mechanisms of nanostructured tribofilms of epoxy-based hybrid nanocomposites," *Wear*, vol. 342–343, pp. 181–188, Nov. 2015, doi: 10.1016/j.wear.2015.08.025.
- [69] S. Bahadur, "The development of transfer layers and their role in polymer tribology," *Wear*, vol. 245, no. 1–2, pp. 92–99, Oct. 2000, doi: 10.1016/S0043-1648(00)00469-5.
- [70] K. I. Alam, A. Dorazio, and D. L. Burris, "Polymers Tribology Exposed: Eliminating Transfer Film Effects to Clarify Ultralow Wear of PTFE," *Tribology Letters*, vol. 68, no. 2, p. 67, May 2020, doi: 10.1007/s11249-020-01306-9.
- [71] G. Yu *et al.*, "The pivotal role of oxygen in establishing superlow friction by inducing the in situ formation of a robust MoS₂ transfer film," *Journal of Colloid and Interface Science*, vol. 594, pp. 824–835, 2021, doi: <https://doi.org/10.1016/j.jcis.2021.03.037>.
- [72] K. Nishimura, K. Masuda, T. Yamamura, Y. Hara, M. Ono, and S. Sasaki, "Transfer Film Composition and Characteristics in Copper-Free NAO Brake Pads," SAE International, Warrendale, PA, SAE Technical Paper 2021-01-1278, Oct. 2021. doi: 10.4271/2021-01-1278.

-
- [73] P. Baskara Sethupathi and J. Chandradass, "Comparative study of different solid lubricants towards friction stability in a non-asbestos disc brake pad," *Industrial Lubrication and Tribology*, vol. 73, no. 6, pp. 897–903, Jan. 2021, doi: 10.1108/ILT-04-2021-0147.
- [74] W. Österle and I. Urban, "Third body formation on brake pads and rotors," *Tribology International*, vol. 39, no. 5, pp. 401–408, May 2006, doi: 10.1016/j.triboint.2005.04.021.
- [75] W. Österle, A. I. Dmitriev, B. Wetzel, G. Zhang, I. Häusler, and B. C. Jim, "The role of carbon fibers and silica nanoparticles on friction and wear reduction of an advanced polymer matrix composite," *Materials & Design*, vol. 93, pp. 474–484, Mar. 2016, doi: 10.1016/j.matdes.2015.12.175.
- [76] J. Ye, H. S. Khare, and D. L. Burris, "Quantitative characterization of solid lubricant transfer film quality," *Wear*, vol. 316, no. 1–2, pp. 133–143, Aug. 2014, doi: 10.1016/j.wear.2014.04.017.
- [77] L. Chang, K. Friedrich, and L. Ye, "Study on the Transfer Film Layer in Sliding Contact Between Polymer Composites and Steel Disks Using Nanoindentation," *Journal of Tribology*, vol. 136, no. 2, Dec. 2013, doi: 10.1115/1.4026174.
- [78] H. Li, Z. Yin, D. Jiang, L. Jin, and Y. Cui, "A study of the tribological behavior of transfer films of PTFE composites formed under different loads, speeds and morphologies of the counterface," *Wear*, vol. 328–329, pp. 17–27, Apr. 2015, doi: 10.1016/j.wear.2015.01.028.
- [79] R. Sebastian, *Advanced in-situ measurements within sliding contacts*, Als Ms. gedr. Kaiserslautern: Inst. für Verbundwerkstoffe, 2014.
- [80] N. X. Randall and J. L. Bozet, "Nanoindentation and scanning force microscopy as a novel method for the characterization of tribological transfer films," *Wear*, vol. 212, no. 1, pp. 18–24, Nov. 1997, doi: 10.1016/S0043-1648(97)00145-2.
- [81] N. X. Randall and A. Harris, "Nanoindentation as a tool for characterising the mechanical properties of tribological transfer films," *Wear*, vol. 245, no. 1, pp. 196–203, Oct. 2000, doi: 10.1016/S0043-1648(00)00479-8.
- [82] V. K. Jain and S. Bahadur, "Material transfer in polymer-polymer sliding," *Wear*, vol. 46, no. 1, pp. 177–188, Jan. 1978, doi: 10.1016/0043-1648(78)90119-9.

- [83] T. W. Scharf and I. L. Singer, "Monitoring Transfer Films and Friction Instabilities with In Situ Raman Tribometry," *Tribology Letters*, vol. 14, no. 1, pp. 3–8, 2003, doi: 10.1023/A:1021942830132.
- [84] T. W. Scharf and I. L. Singer, "Quantification of the Thickness of Carbon Transfer Films Using Raman Tribometry," *Tribology Letters*, vol. 14, no. 2, pp. 137–145, Feb. 2003, doi: 10.1023/A:1021942822261.
- [85] T. W. Scharf and I. L. Singer, "Third bodies and tribochemistry of DLC coatings," in *Tribology of diamond-like carbon films*, Springer, 2008, pp. 201–236.
- [86] R. R. Chromik, C. C. Baker, A. A. Voevodin, and K. J. Wahl, "In situ tribometry of solid lubricant nanocomposite coatings," *Wear*, vol. 262, no. 9, pp. 1239–1252, Apr. 2007, doi: 10.1016/j.wear.2007.01.001.
- [87] R. R. Chromik, A. L. Winfrey, J. Lüning, R. Nemanich, and K. J. Wahl, "Run-in behavior of nanocrystalline diamond coatings studied by in situ tribometry," *Wear*, vol. 265, no. 3–4, pp. 477–489, Jul. 2008, doi: 10.1016/j.wear.2007.11.023.
- [88] G. Meneghetti, A. Terrin, and S. Giacometti, "A twin disc test rig for contact fatigue characterization of gear materials," *Procedia Structural Integrity*, vol. 2, pp. 3185–3193, Dec. 2016, doi: 10.1016/j.prostr.2016.06.397.
- [89] M. Kerridge and J. K. Lancaster, "The stages in a process of severe metallic wear," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 236, no. 1205, pp. 250–264, Aug. 1956, doi: 10.1098/rspa.1956.0133.
- [90] E. Rabinowicz, D. Tabor, and F. P. Bowden, "Metallic transfer between sliding metals: an autoradiographic study," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 208, no. 1095, pp. 455–475, Sep. 1951, doi: 10.1098/rspa.1951.0174.
- [91] E. Rabinowicz and K. V. Shooter, "The Transfer of Metal to Plastics during Sliding," *Proceedings of the Physical Society. Section B*, vol. 65, no. 9, pp. 671–673, Sep. 1952, doi: 10.1088/0370-1301/65/9/303.
- [92] M. D. Placette, S. Roy, D. White, S. Sundararajan, and C. J. Schwartz, "The effect of surface roughness orientation on PEEK (polyetheretherketone) transfer film volume in multi-directional and linear sliding," *Wear*, vol. 426–427, pp. 1345–1353, Apr. 2019, doi: 10.1016/j.wear.2019.01.035.

-
- [93] K. Bashandeh, P. Lan, J. L. Meyer, and A. A. Polycarpou, "Tribological Performance of Graphene and PTFE Solid Lubricants for Polymer Coatings at Elevated Temperatures," *Tribology Letters*, vol. 67, no. 3, p. 99, Aug. 2019, doi: 10.1007/s11249-019-1212-5.
- [94] W. Sun, X. Liu, K. Liu, W. Wang, and J. Ye, "Ultralow wear PTFE composites filled with beryllia and germania particles," *Wear*, vol. 450–451, p. 203270, Jun. 2020, doi: 10.1016/j.wear.2020.203270.
- [95] W. Cui *et al.*, "Role of transfer film formation on the tribological properties of polymeric composite materials and spherical plain bearing at low temperatures," *Tribology International*, vol. 152, p. 106569, Dec. 2020, doi: 10.1016/j.triboint.2020.106569.
- [96] D. J. Smith, *Reliability, Maintainability and Risk: Practical Methods for Engineers Including Reliability Centred Maintenance and Safety-related Systems*. Elsevier Science & Technology, 2005.
- [97] The ImageMagick Development Team, *ImageMagick*. 2021. [Online]. Available: <https://imagemagick.org>

Figures

- Figure 1: Exemplary project workflow. Boxes represent processes, arrows represent workflow and dashed arrows represent main data/information flow. Test setup and conduction are executed in a loop until all tests have been done. Grey: computerized support.26
- Figure 2: Sub-process for the set-up and execution of a single test. Needs to be iterated for each test. Grey: computerized support.....27
- Figure 3: Sub-process for evaluating a single test. Needs to be iterated for each test and test set. Grey: computerized support.27
- Figure 4: Sub-process for report generation. Grey: computerized support.28
- Figure 5: Form for creating new and editing existing materials (left) and Material index table (right, simplified).34
- Figure 6: Bachmann-notation of *Atlas*' test specimen data model. Subclasses are printed light and are connected to their superclasses by empty-headed arrows.37
- Figure 7: *Atlas*' three-level hierarchical data model for individual test specimen, test specimen lots and test specimen classes. For clarity, only the block type subclasses are shown. Corresponding subclasses exist for all other supported test specimen types, see Table 2.37
- Figure 8: Atypical upstream of attribute queries and downstream of attribute data.38
- Figure 9: Test specimen data model for downstream property propagation: all propagatable attributes are present in each level of the hierarchy.....39
- Figure 10: Test specimen index with different type setting for propagated (grey, italic) and non-propagated (upper face, bold) attributes of individual test specimen (here: width, gauge, height and cross section of blocks).....42
- Figure 11: Forms for creating new test specimen in *Atlas*.44
- Figure 12: Left: block on ring-tribometer with four parallelized sets of block/ring-pairs. Rings are mounted to one common shaft that is

driven by a drive. Right: pin on disc-tribometer with four parallelized sets of pin/disc-pairs.	45
Figure 13: Modelling of the relation of wear tests and tribometers by (testing) positions.	46
Figure 14: ERD of <i>Atlas</i> ' data model for wear tests.	48
Figure 15: ERD of the wear test and wear test segment models including their respective subclasses.	49
Figure 16: Polymorphic association between <code>BlockOnRingTest</code> and <code>Ring</code> . <code>Ringable</code> is the name of the polymorphic association between any of the ring-type subclasses of <i>Atlas</i> ' main test specimen classes and subclasses of <code>WearTest</code>	53
Figure 17: Relationship between wear tests, (testing) positions and tribometers.	54
Figure 18: Relationships between projects, wear test sets and wear tests.	54
Figure 19: Form-based creation of wear tests for a four-position tribometer, step 1: general data.	57
Figure 20: Form-based creation of wear tests for a four-position tribometer, step 2: test specimen data.	57
Figure 21: Form-based creation of wear tests for a four-position tribometer, step 3: loading information snippets for validation.	57
Figure 22: Form-based creation of wear tests for a four-position tribometer, step 4.1: define type and number of segments to add.	58
Figure 23: Form-based creation of wear tests for a four-position tribometer, step 4.2: definition of a sequence of six segments that vary in temperature.	58
Figure 24: HTML-view of tests that belong to a sample project, grouped by wear test sets and sorted by test set id, segment index and wear test id.	60
Figure 25: Automatic configuration of a tribometer with <i>Atlas</i> . Grey: computerized support. Method Support is subject to support from the tribometer control software.	62
Figure 26: Form for starting a push workflow for tribometer configuration.	63

- Figure 27: Example of a tab-separated columnar data file from a tribometer. The column width exceeds the viewer tab width at several occasions, resulting in partial shift of columns..... 65
- Figure 28: Example of a YAML-file that contains an *Atlas* data file column descriptor..... 66
- Figure 29: Use of reversal point markers for segmenting data files that represent oscillation segments. 70
- Figure 30: Workflow for uploading measurement data files. Grey: computerized support. *Level of support depends on machine. 74
- Figure 31: Controls for creating a new graph for recorded data (and columnar data derived from it)..... 80
- Figure 32: Form for changing the settings of a recorded data graph. 81
- Figure 33: Exemplary plots. Top left: collection of the COFs of eight individual tests, top right: arithmetic mean, upper and lower confidence of the COFs of 8 individual tests, center left: linear wear rates of 6 different materials, center right: response surface plot (COF vs. normal force and sliding speed), data points represent arithmetic means of a set of 12 individual tests, bottom left: multi-segment of recorded data, bottom right: COF and linear wear rate versus sliding speed for a set of eight. 83
- Figure 34: Details page for an individual wear test. On this page, all relevant test parameters, associated objects (tribometer, test set, project) can be viewed. Furthermore, actions like evaluating or resetting one or all segments of a test can be executed, data files can be downloaded, plots of measured data can be viewed (not shown) and results and parameters of auxiliary investigations, like profilometer scans or transfer film luminance analysis (see Section 5) can be viewed. 84
- Figure 35: View for the joint evaluation of segments of different tests that belong to the same wear test set. All segments have the same segment index. This overview helps the user with assessing the validity of the evaluation of individual segments. The view offers control elements for adjusting the ranges in which various properties should be calculated. Furthermore, it provides basic

results for each segment, like linear wear rate and coefficient of friction. Finally, it provides information about the overall validity status of the evaluation as well as control to set this status.	85
Figure 36: Project-level view of wear tests. Tests that belong to a common set are grouped together and groups are sorted by segment index in ascending order. In contrast to test-level and set-level view, no time resolved plots of measured or derived quantities are shown. Instead, test-level aggregates are shown. Typical examples for this are arithmetic means of various temperatures or the coefficient of friction. The list of displayed columns can be defined in a wide range and on a per-project basis. In addition to test-level aggregates, set-level aggregates are shown. These are mostly statistical data like arithmetic mean, standard deviation, confidence interval or the result of an automatic outlier test. In addition, validity data are shown and can be modified.	86
Figure 37: Form for customizing the content of automatically generated reports.	88
Figure 38: Schematics of the experimental setup for a block on ring experiment: diagonal view with longitudinal section through the light dome (left) and section of the whole test setup showing typical ray traces (right)	102
Figure 39: Wear debris patches located on the wear track are discretized on the camera sensor.	103
Figure 40: Transition from short exposure image to circumferentially averaged image by exposure time matching (top). Luminance change due to circumferential averaging (bottom).	105
Figure 41: Exemplary plot of Δl_{rel} as a function of lateral wear track position x and elapsed test time t (left) and Exemplary plot of $N_{roi}/reft$ (top right) and of Δl_{relt} (right bottom).	111
Figure 42: Example of local increase or decrease of brightness due to polishing, depending on angle, camera position and ring curvature.	112

Figure 43: Post-test image of wear tracks 1 and 2: while track 1 lacks any polishing, track 2 is polished at its center.	112
Figure 44: (Uncorrected) changes of relative luminance for wear tracks 1 and 2. The markers indicate the data points that correspond to the situation shown in the post-test image in Figure 43.....	113
Figure 45: Effect of wear track polishing on ray traces. Top left: diffuse reflection in the center region of the image, some light reaches the camera because diffuse reflection emits light into all directions, including the direction of the camera. Top right: exclusive specular reflection in the center region of the image due to polishing, no light reaches the camera anymore. Bottom left: diffuse reflection below the center region of the image, some light reaches the camera. Bottom right: specular reflection due to polishing of the wear track – now much more light than with diffuse reflection reaches the camera.....	114
Figure 46: Exemplary definition of the evaluation regions needed for automated polishing detection. 1: roi, 2: ref, 3: pol, 4: pol-ref.	115
Figure 47: x,t -plots for exemplary wear tracks 1 (top) and 2 (bottom). The color scale's upper limit represents the utilized α value. Therefore, white color indicates the exclusion of the respective l data from the calculation of $\Delta L_{\text{rel}}(t)$	116
Figure 48: Polishing filtered version of the change of relative luminance (left) and polishing ratios for wear tracks 1 and 2.	118
Figure 49: Workflow of TLA evaluation with <i>Atlas</i> . Grey: computerized support, white: manual.	121
Figure 50: Software tool provided by <i>Atlas</i> for determining TLA evaluation regions ("segmentation") and for starting the evaluation.	122
Figure 51: Example of an automatically generated report, page 1.	146
Figure 52: Example of an automatically generated report, page 2.	147
Figure 53: Example of an automatically generated report, page 3.	148

Tables

Table 1: Historical and existing electronical data systems for material and tribological data, reproduced from [9], expanded	4
Table 2: Test specimen classes modelled in <i>Atlas</i> and their specific attributes.	35
Table 3: Wear test segment subclasses and their domain-specific attributes.	50
Table 4: Estimates for human error rates of elementary processes, reproduced from [52] with modification.	55
Table 5: Classification of methods for transfer film analysis based on five selected issues.	98

Code listings

Code listing 1: Deciding whether water and ethanol are frozen at -14.7 °C using functional programming in Ruby (=> denotes return values).	14
Code listing 2: Deciding whether a liquid is frozen or not, OOP style.	15
Code listing 3: Attributes and encapsulation – freezing temperatures defined at object creation cannot be changed later.....	15
Code listing 4: Class inheritance in Ruby	16
Code listing 5: Method overriding	17
Code listing 6: Surface as an aggregation class whose instances propagate their area property.....	18
Code listing 7: Example for the automatic generation of attribute getters and setters for subclasses of ActiveRecord::Base.....	39
Code listing 8: Implementation of atypical property propagation for width.	40
Code listing 9: Implementation of inverse property propagation.	41
Code listing 10: Property propagation using AttributePropagation.	42
Code listing 11: Implementation of the LabelLable module and its use in test specimen-related superclasses.....	43
Code listing 12: Association between tribometer and testing position(s).	45
Code listing 13: A tribometer’s number of test positions is a propagated attribute.	46
Code listing 14: Declaration of the relationship between tests and test segments.	49
Code listing 15: Implementation of the duration of a wear test segment as a derived attribute using UAP (handling of units omitted).	51
Code listing 16: Declaration of the relationship between block on ring- and ball on plate-tests and their respective test specimens using class methods provided by ActiveRecord.	52
Code listing 17: Declaration of the inverse relation between block on ring wear tests and blocks in the Block, BlockLot, BlockClass classes.....	53
Code listing 18: Definition of singleton-methods for two tribometer instances.....	68

Code listing 19: Dynamic definition of a singleton-method that transforms a data file for wear_test located at file_path to standard format.	68
Code listing 20: Association between wear tests and data files.	72
Code listing 21: Using datafile instances to load data from disk and to get columnar data for measured ambient humidity.....	72
Code listing 22: Association between wear tests and asset folders.	73
Code listing 23: AssetFolder	73
Code listing 24: Implementation of type specific calculations of the same information using specialization.....	76
Code listing 25: Implementation of a method for performing elementary calculations during the evaluation of wear test segments. For clarity, only one such calculation is shown. Other calculations can be included by adding new branches to the case statement.	78
Code listing 26: Partial overriding of the compute method to reflect segment-type specific domain logic without overriding all functionality of the superclass' implementation.	78
Code listing 27: Calculation of columnar data for the coefficient of friction, its storage in a "proc"-type data file object and persistence of the data file object to disk.	79
Code listing 28: Implementation of the valid method on the test segment-level.	87
Code listing 29: Concurrent processing of images using mutual thread exclusion.	124

Appendix

TRIBOFORCE PA66 C0200 UNGEFAERBT vs. 100Cr6		12/20/2021, 02:03 pm UTC
$p = 3.0 \text{ MPa}$, $v = 1.5 \text{ m/s}$, $\vartheta_{cb} = \text{SAT} (23 \text{ }^\circ\text{C})$		
Test report		
Material name	TRIBOFORCE PA66 C0200 UNGEFAERBT	
Material lot	Lot 1311: 0000900743	
Manufacturer	Brenntag GmbH	
Test parameters		
Standard	Block on ring test	
Time period	07/04/2019, 01:15 pm UTC - 07/05/2019, 01:11 pm UTC	
Sliding orientation	Parallel	
Loading parameters		
Loading	3.0 MPa	
Type of motion	Unidirectional sliding	
Sliding speed	1.5 m/s	
Sliding distance	32400.0 m	
Set temperature	SAT (23 °C)	
Counter body		
Name	100Cr6	
Lubrication		
Lubricant name	dry sliding	
IVW GmbH · Erwin-Schrödinger-Str. 58 · 67663 Kaiserslautern · Germany · www.ivw-uni-kl.de		
		1

Figure 51: Example of an automatically generated report, page 1.

TRIBOFORCE PA66 C0200 UNGEFAERBT vs. 100Cr6				
$p = 3.0 \text{ MPa}$, $v = 1.5 \text{ m/s}$, $\vartheta_{cb} = \text{SAT}$ (23 °C)				
12/20/2021, 02:03 pm UTC				
individual tests numeric results				
Wear test ID	Linear wear rate w_t	Specific wear rate w_s	COF dyn., ss	Counter body temperature
Test segment	[$\mu\text{m/h}$]	[$10^{-6} \text{ mm}^3/\text{Nm}$]	[1]	ϑ_{cb} [°C]
547643.1	6.7	0.42	0.21	39
547644.1	8.0	0.49	0.20	46
547645.1	6.5	0.40	0.20	43
547646.1	6.4	0.40	0.25	38
547647.1	7.7	0.48	0.21	39
547648.1	7.9	0.49	0.21	46
547649.1	7.7	0.48	0.21	44
547650.1	5.2*	0.32*	0.24	39
$\bar{x} \pm \Delta x$	7.3 ± 0.6	0.45 ± 0.04	0.22 ± 0.01	42 ± 3
σ, n	0.7, 7	0.04, 7	0.02, 8	3, 8
$c = \sigma/\bar{x}$	0.095	0.095	0.079	0.078
$(\bar{x} \pm \Delta x)$	7.0 ± 0.8	0.43 ± 0.05	0.22 ± 0.01	42 ± 3
(σ, n)	1.0, 8	0.06, 8	0.02, 8	3, 8
$(c = \sigma/\bar{x})$	0.141	0.141	0.079	0.078

* = outlying observation (i.e. automatically invalidated)
** = manually invalidated
(...) = parameter incl. outlying observations
 \bar{x} = arithmetic mean, Δx = confidence, $\bar{x} \pm \Delta x$ = confidence interval
 σ = standard deviation of sample, n = sample size, c = coefficient of variance

IVW GmbH · Erwin-Schrödinger-Str. 58 · 67663 Kaiserslautern · Germany · www.ivw-uni-kl.de 2

Figure 52: Example of an automatically generated report, page 2.

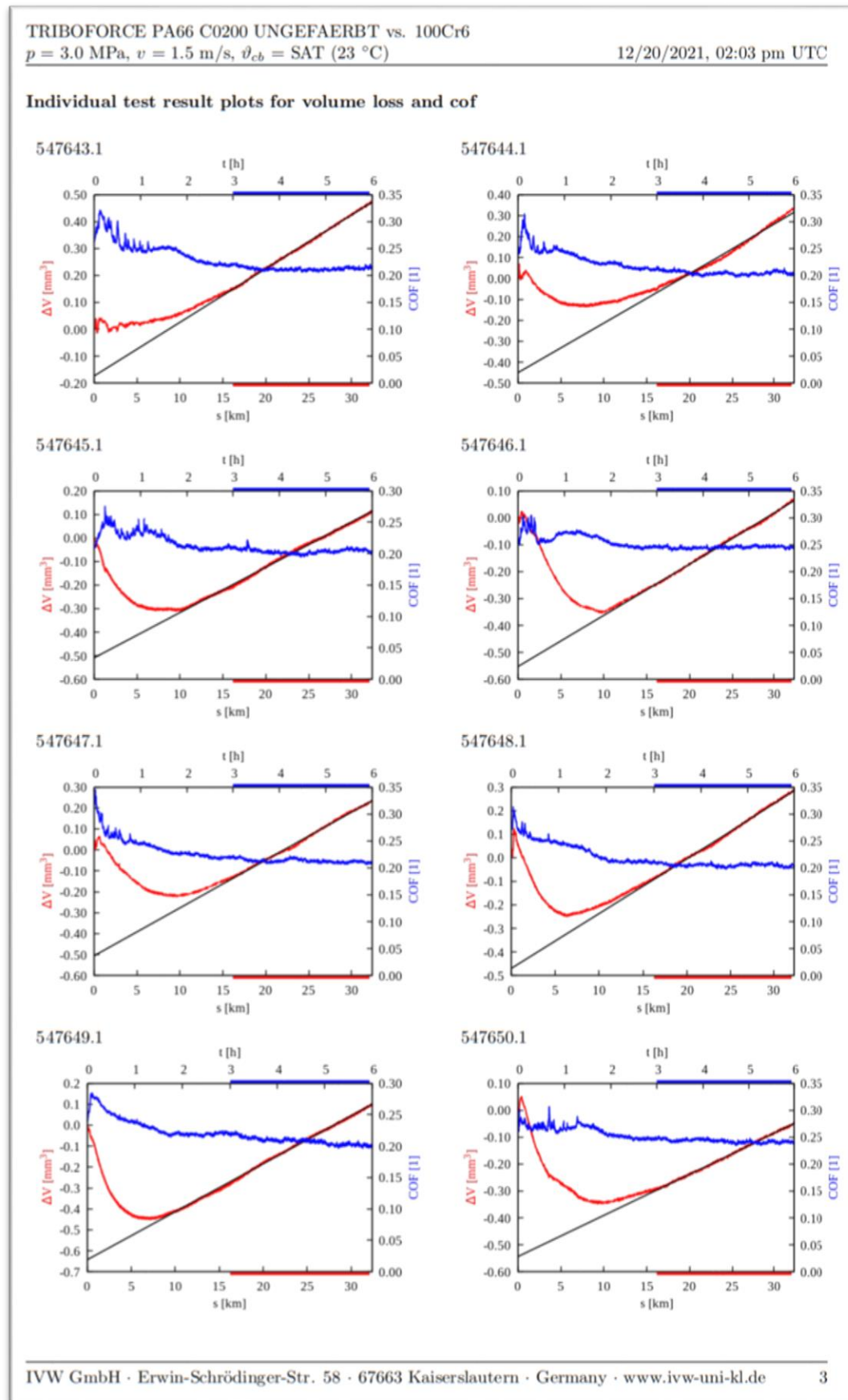


Figure 53: Example of an automatically generated report, page 3.

Publications and conference contributions

Peer-reviewed publications

1. N. Knör, **A. Gebhard**, F. Hauptert, A.K. Schlarb, Polyetheretherketone (PEEK) nanocomposites for extreme mechanical and tribological loads, *Mechanics of Composite Materials* 42, 199-206 (2009), DOI: 10.1007/s11029-009-9071-z
2. **A. Gebhard**, T. Bayerl, A.K. Schlarb, K. Friedrich, Galvanic corrosion of polyacrylnitrile (PAN) and pitch based short carbon fibres in polyetheretherketone (PEEK) composites, *Corrosion Science* 51, 2524-2528 (2009), DOI: 10.1016/j.corsci.2009.05.051
3. **A. Gebhard**, T. Bayerl, A.K. Schlarb, K. Friedrich, Increased wear of aqueous lubricated short carbon fiber reinforced polyetheretherketone (PEEK/SCF) composites due to galvanic fiber corrosion, *Wear* 268, 876-876 (2010), DOI: 10.1016/j.wear.2009.11.018
4. H. Hunke, N. Soin, **A. Gebhard**, T. Shah, E. Kramer, K. Witan, A.A. Narasimulu, E. Siores, Plasma modified Polytetrafluoroethylene (PTFE) lubrication of α -olefin-copolymer impact-modified Polyamide 66, *Wear*, 338-339, 112-132 (2015), DOI: 10.1016/j.wear.2015.06.003
5. **A. Gebhard**, B.C. Jim, Photo-optical luminance analysis of transfer films: Measurement principle, data analysis and result plotting, *Tribology International* 153, 106626 (2021), DOI: 10.1016/j.triboint.2020.106626
6. **A. Gebhard**, B.C. Jim, Formation, stability and degradation of transfer films formed by polyphenylene sulfide (PPS) and its composites in dry sliding against steel, *Wear*, in press, DOI: 10.1016/j.wear.2022.204343

Non-peer reviewed publications

7. **A. Gebhard**, N. Knör, F. Hauptert, A.K. Schlarb, Nano particle reinforced high performance thermoplastic material for extreme tribological loads in automobile manufacture, *Tribologie und Schmierungstechnik* 55 (4), 28-32, (2008)
8. **A. Gebhard**, S. Emrich, M. Kopnarski, F. Hauptert, A.K. Schlarb, Polyetheretherketone for boundary friction stressed multi-layer rolled sliding bearings in diesel injection pumps, *Tribologie und Schmierungstechnik* 55 (5), 31-55 (2008).

9. **A. Gebhard**, F. Hauptert A.K. Schlarb, Chapter 18: Development of nanostructured slide coatings for automotive components, in: K. Friedrich, A.K. Schlarb (Eds.), Tribology of Polymeric Nanocomposites (Second Edition), Butterworth-Heinemann, Oxford, 2013, 619–648. DOI:10.1016/B978-0-444-59455-6.00018-0

Data publications

10. **A. Gebhard**, B.C. Jim, Data for: Photo-optical luminance analysis of transfer films: Measurement principle, data analysis and result plotting, Mendeley Data, V2, DOI: 10.17632/y9dgyhx99w.2
11. **A. Gebhard**, M. Fickert, Data for: Improved design process of dry-running radial plastic plain bearings by coupling laboratory tests and component simulation, Mendeley Data, V1, DOI: 10.17632/549yvgmvyk.1
12. **A. Gebhard**, M. Fickert, Data from Block-on-Ring Wear Tests of Victrex FG340 in Dry Sliding versus AISI 52000 steel, Mendeley Data, V1, DOI: 10.17632/nbz7hs4ckn.1

Conference Contributions

13. **A. Gebhard**, B.C. Jim, B. Wetzel, Moderne Prüftechnik in der Tribologie: Zeitauflösende optische Detektion und Quantifizierung von Transferfilmen in Kunststoff-Stahl-Gleitkontakten, vdi-Fachkonferenz „Hochleistungs-Kunststoffzahnräder“ 2018, München
14. **A. Gebhard**, B.C. Jim, B. Wetzel, Advanced testing methods for polymer transfer films in sliding contacts, Wear Resistant Plastics 2018, Düsseldorf
15. **A. Gebhard**, B.C. Jim, B. Wetzel, Neues Verfahren zur quantitativen in situ-Erfassung von Transferfilmen in Kunststoff-Metall-Gleitkontakten, 59. Tribologie-Fachtagung, 2018, Göttingen
16. R. Walter, **A. Gebhard**, M. Gurka, T. Huber, B. Wetzel, Verbesserung der Gleitverschleißigenschaften durch Schmierstoff-zuführung in einer porösen, additiv gefertigten Kunststoffstruktur, 59. Tribologie-Fachtagung, 2018, Göttingen
17. B.C. Jim, **A. Gebhard**, B. Wetzel, In-situ measurement of transfer films by a novel optical method using the example of polymer-metal sliding contacts, Polytrib 2018, 2018, Portoroz (Slowenien)

-
18. **A. Gebhard**, ATLAS – an automated and highly integrated information management system for tribology laboratories, 60. Tribologie-Fachtagung, 2019, Göttingen
 19. B.C. Jim, **A. Gebhard**, B. Wetzel, On the kinetics and stability of transfer films in polymer/metal sliding pairings, 60. Tribologie-Fachtagung, 2019, Göttingen
 20. B.C. Jim, **A. Gebhard**, B. Wetzel, R.J. Robin, Novel method for the quantitative in-situ detection of transfer films in polymer-steel sliding contact, SPE Automotive Composites Conference & Exhibition 2019, Detroit, Michigan, USA
 21. **A. Gebhard**, B.C. Jim, Photo-optical transfer film detection and its application to material design, 30 years IVW – Anniversary Colloquium, 2021, Kaiserslautern
 22. M. Fickert, **A. Gebhard**, Verbesserte Auslegung trockenlaufender Radialgleitlager aus Kunststoff durch die Kopplung von Laborversuch und Bauteilsimulation, 61. Tribologie-Fachtagung, 2021, Göttingen
 23. M. Fickert, **A. Gebhard**, Improved design process of dry-running radial plastic plain bearings by coupling laboratory tests and component simulation, 61. Tribologie-Fachtagung, 23rd International Colloquium Tribology, 2022, Stuttgart/Ostfildern

Supervised Theses

1. B.C. Chim, Entwicklung einer (teil-)automatisierten Methode zur Struktur-
aufklärung an Reibungs- und Verschleißoberflächen, Studienarbeit, 2020
2. T. Assmus, Innovative Transferfilm-Luminanzanalyse im klimatisierten
Stift/Scheibe-Versuch, Masterarbeit, 2020
3. E. Kuksa, Untersuchung der temperaturabhängigen Bildungskinetik und
Stabilität von Transferfilmen in Polymer-Metall-Gleitkontakten, Bachelorar-
beit, 2020
4. S. Aumann, Gleit- und Schwingverschleiß ausgewählter Mehrschichtver-
bungleitlager-Halbzeuge unter trockenen und geschmierten Bedingungen,
Studienarbeit, 2007
5. M. Fickert, Konstruktive Optimierung eines Block-auf-Ring-Verschleißprüf-
stands und Implementierung einer räumlich und zeitlich hochaufgelösten
Verschleißfassung, Studienarbeit, 2007
6. T. Bayerl, Verschleißverhalten von kohlenstofffaserverstärktem PEEK in
wässriger Umgebung, Diplomarbeit, 2006

Curriculum Vitae

Personal data

Name Andreas Gebhard
Place of birth Ludwigshafen am Rhein, Germany
Nationality German

Work experience

07/2017 – present Manager Competence Field „Tribology“, Leibniz-Institut für
 Verbundwerkstoffe GmbH, Kaiserslautern, Germany
07/2010 – 12/2020 Owner-manager of Tribologic GmbH
07/2008 – 06/2017 Head of Research and Development of Tribologic GmbH
07/2005 – 06/2008 Research Associate, Institut für Verbundwerkstoffe GmbH,
 Department of Materials Science
04/2005 – 06/2005 Member of the Graduate College „Non-linear Optics and Ultra-
 fast Physics“, Department of Physics, University of Kaiserslau-
 tern
02/2004 – 03/2005 Research Associate, Department of Chemistry, University of
 Kaiserslautern

Education

01/2019 – PhD, Leibniz-Institut für Verbundwerkstoffe GmbH, Kaisers-
 lautern
01/2004 Diploma Thesis „Organic photorefractive glasses with high
 transparency“
04/2002 – 07/2002 Erasmus Exchange Program, The University of Edinburgh,
 Edinburgh, Scotland
10/1998 – 01/2004 Diploma Program Chemistry, University of Kaiserslautern

ISSN 1615-021X
ISBN 978-3-944440-51-4