FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Model-Driven Benchmark Data Generation for Digital Preservation of Webpages

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering and Internet Computing

eingereicht von

Clemens Sauerwein, BSc
Matrikelnummer 0716649

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dr. Andreas Rauber
Mitwirkung: Dr. Christoph Becker

Wien, 02.12.2013

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Model-Driven Benchmark Data Generation for Digital Preservation of Webpages

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Clemens Sauerwein, BSc
Registration Number 0716649

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dr. Andreas Rauber
Assistance: Dr. Christoph Becker

Vienna, 02.12.2013

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Clemens Sauerwein, BSc
Lahntalweg 23, 6020 Innsbruck

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasser)

Abstract

Digital Preservation (DP) is the process of keeping digital information accessible and usable in an authentic manner for a long term. Preservation activities are used to guarantee long term and error free accessibility of data regardless of technological change. Different approaches based on continuous transformation of data are used to perform these preservation activities. Several tools exist for the execution of these activities.
Digital objects have significant properties which must be preserved during the transformations. To evaluate these preservation activities information about these characteristics (e.g. structure, size) are necessary. The annotations of digital objects with this information are used as ground truth.
A benchmark data set can be formed with real world data but the verification of the properties has to be done manually. Every automatic analysis is based on the correct interpretation of an analysis program (e.g. characterization tool). Due to the fact that these programs must be evaluated there is a profound lack of annotated benchmark data in Digital Preservation. For this reason the evaluation and improvement of digital preservation approaches and tools is hindered.
This thesis introduces a model driven benchmark data generation framework with the purpose of automatic generation of benchmark data with corresponding ground truth. The system uses the Model Driven Architecture (MDA) as underlying concept which facilitates the usage of well-known model driven engineering tools and frameworks. Instead of analyzing existing benchmark data collections of computer science it generates the benchmark data sets referred to property distributions of different kinds of documents (e.g. webpages). The framework specifies ground truths for the Platform Independent and Platform Specific Models of the generated benchmark data. These ground truths together with the benchmark data are used for evaluation.
The model driven benchmark data generation framework is evaluated by generating benchmark data for testing preservation action tools for web pages. They are widely used and a complex challenge in digital preservation settings. We define a Platform Independent and a Platform Specific Model for representing webpages and demonstrate how benchmark data can be created with these.

Kurzfassung

Die digitale Langzeitarchivierung ist jener Prozess, der die langfristige Zugänglichkeit und authentische Benutzbarkeit von digitalen Informationen gewährleistet. Langzeitarchivierungsaktivitäten werden verwendet um eine langfristige und fehlerfreie Datenspeicherung unabhängig von technologischen Veränderungen zu gewährleisten. Die verschiedenen Ansätze zur kontinuierlichen Transformation von Daten werden verwendet, um die Langzeitarchivierungsaktivitäten durchzuführen. Dazu gibt es eine Vielzahl von Tools.
Digitale Objekte haben signifikante Eigenschaften, welche während der Transformationen erhalten bleiben müssen. Um die Langzeitdatenarchivierungstätigkeiten zu testen, werden Informationen über die Eigenschaften (z.B. Struktur) benötigt. Die Annotation von Digitalen Objekten mit diesen Informationen wird als Grundwahrheit verwendet.
Daten der realen Welt lassen sich zu einem Testdatenkorpus zusammentragen, jedoch müssen diese manuell verifiziert werden. Jede automatische Analyse basiert auf der korrekten Interpretation eines Analyseprogrammes (z.B. Charakterisierungstool). Aufgrund der Tatsache, dass diese Programme getestet werden müssen, gibt es einen grundlegenden Mangel an Testdaten mit entsprechenden Annotationen. Aus diesem Grund werden die Evaluierung und die Verbesserung von bestehenden Ansätzen und Tools erschwert.
Diese Arbeit führt ein modellgetriebenes Testdatengenerierungsframework ein. Das Ziel des Systems ist die automatische Generierung von Testdaten mit entsprechenden Grundwahrheiten. Als zugrundeliegendes Konzept nützt das Framework die modellgetriebene Architektur (eng. Model Driven Architecture). Diese ermöglicht die Nutzung einer Vielzahl von bekannten Tools und Frameworks der modellgetriebenen Softwareentwicklung. Anstatt der Analyse existierender Testdatensammlungen der Informatik, erzeugen wir Webseiten, basierend auf statistischen Verteilungen von Eigenschaften von Dokumenten (z.B. Webseiten). Das Framework spezifiziert Grundwahrheiten für das plattformunabhängige und plattformspezifische Modell der generierten Testdaten. Diese Grundwahrheiten zusammen mit den Testdaten werden für die Evaluierung verwendet. Das modelgetriebene Testdatengenerierungsframework wird durch die Erzeugung von Webseiten, die weitverbreitet und komplex sind, evaluiert. Wir definieren ein plattformunabhängiges und ein plattformspezifisches Modell für die Repräsentation von Webseiten und demonstrieren wie Testdaten damit erzeugt werden.

Contents

CHAPTER 1

Introduction

All over the world there exist a large amount of valuable Digital Objects which contain cultural, scientific, educational and other kinds of information. Most of them are created digitally or converted from existing analogue resources into digital form. These digital objects include images, texts, audio files, videos, webpages, software and a wide range of other formats [1] [2].
The challenge is to ensure access to these Digital Objects for as long as possible otherwise the information would get lost forever. Digital Preservation is the process of maintaining objects accessible and reusable in an authenticated manner for a long term into the future. Thereby it goes beyond the limits of media failure or technological change [3]. Digital Preservation offers diverse approaches (e.g. emulation, migration) to ensure long term access to digital objects.
After applying these approaches the performance of the digital objects play an important role. H. Heslop et al. states that "*performance is what is rendered to the screen or to any other output device* [4]." The performance of a digital object should be always nearly the same regardless of the applied approaches or strategies. For the decision about the correctness of them the digital objects have to be evaluated. This evaluation is facilitated if there exists a ground truth or annotation for every digital object. A ground truth holds information about the significant properties (e.g. structure) of a digital object and specifies comparable metrics for experimentation.
The goal of this thesis is to find a way to generate annotated benchmark data collections for the evaluation of digital preservation approaches. While other research areas in computer science have benchmark data collections one of the main problems in Digital Preservation is a profound lack of annotated benchmark data. Several researches in the field of Digital Preservation state the necessity for it. For example C. Becker et al. states that " *annotated benchmark data are needed to support the objective comparison of new approaches and quantify the improvements over existing techniques [5].*" [6] and [7] outline that it is necessary to have metrics for evaluating the correctness of tools (e.g. characterization tools). Therefore Digital Preservation needs benchmark data collections with corresponding ground truth.
The challenge of this thesis is to find, implement and evaluate a new approach to generate benchmark data with corresponding ground truth for Digital Preservation. The ground truth should support the traceability of features, describe the structure and properties of a digital object.

Moreover the solution should work as automated as possible and should guarantee scalability.
Within this thesis we follow a new approach of benchmark data generation which is different from previous. Instead of analyzing existing data collections we generate them from scratch. Thereby we focus on generating webpages with corresponding ground truth. As the fundament of our benchmark data generation we use statistically distributions of webpage features. These statistics indicate how often a certain feature appears within a webpage. Based on these values we generate our webpages with corresponding ground truths.
As development methodology of our generation framework we use the Model Driven Architecture which facilitates the usage of well-known model driven engineering tools and frameworks. Therefore we look which model driven engineering technology fits best in our use case. To apply this methodology we must define several information models. The usage of information models to represent the Platform Independent (PIM) and the Platform Specific Model (PSM) is crucial.
After developing our model driven benchmark data generation framework we will do experiments with it. We will generate a few pages based on a sample feature distribution. The results of our benchmark data generation experiment will be evaluated. On the one hand we will precisely analyze whether the systems generates the expected webpages and on the other hand we will do a closer look on the execution time and the scalability of our system.
The Thesis is structured as follows:
Chapter 2 gives an overview of Digital Preservation approaches, standards and projects which deal with testing and evaluating them. It describes the challenges which appear while creating benchmark data and it presents existing benchmark data collections in the fields of Digital Forensics, Information Retrieval and Machine Learning. It gives an introduction to Model Driven Architecture (MDA) and its tools, frameworks and technologies. Moreover we give an introduction to web page adaption which adapts existing webpages based on the platform specific needs of a client (e.g. browser, mobile device).
Chapter 3 introduces the Model Driven Benchmark Data Generation Framework. It gives an overview of the system. In a further step the models and processes are explained. It also gives insights to the technical realizations.
Chapter 4 shows tests with focus on scalability and correct functionality of the system. It explains sample transformations and shows their capabilities. Finally, this chapter evaluates the system and shows its limitations.
The last chapter summarizes the thesis and shows how the results can be applied for future research.

CHAPTER 2

Related Work

This chapter starts with an introduction about Digital Preservation, its current state of research and approaches. It presents several projects dealing with testing and evaluating of Digital Preservation strategies.
We also focus on the topic of benchmark data and show existing benchmark data collections which are designed for Digital Forensics, Information Retrieval and Machine Learning. We summarize and outline the most important observations about them.
In a further step we introduce the Model Driven Architecture which is the system's underlying concept. We explain all the components of the Model Driven Architecture (MDA) and evaluate it by showing its advantages and disadvantages.
We introduce the Eclipse Modeling (EMF) framework which is designed for model driven engineering. Thereby we explain and evaluate several technologies and tools which support it.
Finally, we give an introduction to web page adaption. Thereby a webpage is seen as some kind of generic infomrmation which must be transformed in a platform specific representation. These transformations deliver an optimized version of web content for a specific device or browser.

2.1 Digital Preservation

"Heritage is explained in UNESCO documents as our legacy from the past, what we live with today, and what we pass on to future [1]." Heritage can be seen as information that has cultural, historical, archaeological, scientific or anthropological value [1]. Throughout history writing has guaranteed the steady transmission of information. For example monks interpreted and copied books in the Middle Ages. It can be seen as evolution of information [8]. Today we can digitize almost every kind of information and store them as digital materials. Digital materials include texts, images, videos, software and a lot of other formats.
P. Conway states that *"our capacity to record information has increased exponentially over time while the longevity of media to store the information has decreased equivalently [8]."* The same applies to the longevity of file formats. Longevity in context of information means that

the information should be accessible and usable in an authenticated manner for a long term. T. Kuny shows in [9] the following challenges which have to be faced to guarantee accessibility:

- A lot of digital information is lost forever because it wasn't archived properly or resides in a format which cannot be accessed anymore. It would be too expensive and complex to rescue or recover it.
- There is a large amount of Digital Information coming to libraries and archives. It is necessary to find and apply methods for organizing the information.
- Technology in Computer Science is changing rapidly. Therefore technologies are replaced through new ones constantly. This fact creates an unstable and unpredictable environment for continuance of hardware and software.
- A lot of documents and media formats have their own hardware and software dependencies. If the software or hardware which can interpret the media formats doesn't exist anymore the information gets lost forever.
- Money for libraries and archives continue to decrease. Thus, there isn't enough money for preservation activities.
- Licenses of intellectual properties ensure that many materials never make it into a library collection for preservation.
- The challenge of preservation of digital objects is not primarily a technological one it is also a sociological one. The market of technologies and products ensure instability of hardware and software because the obsolescence should promote the business.

These challenges point out that long term accessibility is a non-trivial topic. There exist a lot of more challenges in its context.
The Digital Preservation Coalition defines that *"Digital Preservation refers to the series of management activities necessary to ensure continued access to digital materials* [...] *beyond the limits of media failure or technological change. Those materials may be records created during day-to-day business of an organization, "born digital" materials created for a specific purpose (e.g. teaching resources), or the products of digitization projects [3]."*

2.1.1 State of Research

There are a lot of projects, organizations and tools dealing with Digital Preservation. The following section outlines a few of them but does not provide a full overview.
Building of organizations and research networks has been one of the topics since the beginning of research in Digital Preservation. The purpose of these scientific communities was to raise awareness for the topic of long term preservation and its importance for the future. Moreover these communities should foster the scientific exchange and discussion. One of the first communities was the DELOS [1] network for integrating and coordinating activities of research teams.

[1] http://www.delos.info

In 2009 the DELOS Network stopped their work and some DELOS activities have been carried on by the DL [2] community.
However, still active organizations are the DPC [3], NESTOR [4] or OPF [5]. All of them are facilitating the scientific exchange and partnerships between their members. These close collaborations are used for developing digital preservation approaches and solutions.
There are a lot of tools (e.g. BagIt [6], Plato [7], JHOVE [8]) and projects which support the long term preservation. Traditionally the focus was on preserving office documents and images. Nowadays research is dealing with more complex digital formats. The TIMBUS [9] project develops activities, processes and tools to ensure long term access to business processes. The LiWA [10] project deals with preserving web content in a web archive. Other examples are the Blogforever [11] project for digital preservation of blogs or the wf4Ever [12] project for the preservation of scientific experiments in data intensive science.
Another topic in Digital Preservation is scalability of tools and methods. They should operate on big heterogeneous realistic collections of digital objects. To handle these large collections a high degree of automations is important. The SCAPE [13] project deals with this topic with focus on developing *"open source platform that orchestrates semi-automated workflows for large-scale, heterogeneous collections of complex digital objects [13]."* The ENSURE [14] project deals with this topic with focus on the costs and values of different solutions.
As in other areas of Computer Science, standardization plays an important role.The International Organization of Standardizatio (ISO) defines a reference model for an Open Archival Information System (OAIS). It is defined in the ISO14721:2012 [10] Standard from August 2012. It defines an archive as an organization where humans and systems work together to provide long term access to preserved data. It doesn't provide a concrete implementation it declares only a model how such a system should look like.
A lot of approaches in digital preservation use metadata to guarantee the long term access to digital objects. For this reason the PREMIS [15] project defines a standard for metadata.
Audit and Certification stands in connection with standardization and is a further topic in Digital Preservation. CASPAR [16] and TRAC [17] are dealing with this topic. TRAC provides tools certification of digital repositories. It establishes documentation requirements required for audit

[2]http://www.dlorg.eu/
[3]http://www.dpconline.org
[4]http://www.langzeitarchivierung.de
[5]http://www.openplanetsfoundation.org
[6]http://www.digitalpreservation.gov/tools
[7]http://www.ifs.tuwien.ac.at/dp/plato/intro.html
[8]http://jhove.sourceforge.net/
[9]http://www.timbusproject.net/
[10]http://www.liwa-roject.eu/
[11]http://www.blogforever.eu/
[12]http://www.w4ever-project.org/
[13]http://www.scape-project.eu/
[14]http://ensure-fp7-plone.fe.up.pt/site/
[15]http://www.loc.gov/standards/premis/index.html
[16]http://www.casparpreserves.eu/
[17]http://www.crl.edu/archiving-preservation/digital-archives/metrics-assessing-and-certifying-0

and declares a process for certification.

2.1.2 Approaches to Digital Preservation

Digital Preservation consists not only from secure storage and management it also provides actions to digital objects to guarantee their continued access. Migration and Emulation are the most common approaches [11].

Migration

Garett et al. defined migration as *“the periodic transfer of digital materials from one hardware/software configuration to another or from one generation of computer technology to a subsequent generation [12].”*
Some Libraries are using migration to ensure continued access to the digital materials. Using this approach is costly and estimating the costs would be preferable. In [13] a model for cost estimation is explained. It is based on the Open Archive System (OAIS) [10]. It gives the possibility to calculate the costs of different migration scenarios. For migration services, quality assurance is a crucial topic. C. Becker et al. introduce in [14] a preservation action monitoring infrastructure which provides quality measurement of migration web services. Thereby tools are monitored and the quality of the results are measured.
F. Luan et al. show in [15] that the quality of metadata in context with migration matters. They introduce quality requirements for migration metadata which they derive from common preservation metadata schemas. They make a case study to validate the correctness and usefulness of theses quality requirements [15].

Emulation

S. Sawant states that *“the more complex a digital object is, the more loss will be in its migration to new formats and generation of hardware or software [16].”* It leads to the assumption that for some cases emulation might be better. *“Emulation is the process of recreation of the hardware and software environment required to access a resource [16].”*
As stated in [17] one of the key problems is to provide a wide range of user access to an ancient environment. Therefore K. Rechert et al. introduced the GRATE (Global Remote Access to Emulation) an abstract architecture for emulation services over computer networks without installing additional software components. The user needs only a connection to the network to access the remote user interfaces for emulation and the whole emulation process is done by it. As mentioned C. Becker et al. introduce in [14] an infrastructure to evaluate migration web services. C. Becker et al. provide also a possibility to measure the quality of remote emulation as provided by GRATE.
The KEEP [18] project is developing emulation services for static and dynamic digital objects such as texts, sound, images, multimedia documents and so on. The goal of this project is the development of tools for accessing a wide range of digital objects using emulation tools or migrate these digital objects to another environment.

[18]http://www.keep-project.eu/

Another challenge within migration is that availability of tools for older formats is often limited. In [18] K. Rechert et al. use original applications in an emulated environment to access these tools. In a further step K. Rechert et al. transform the outputs of these tools into formats which can be accessed today.

2.1.3 Testing and evaluating of Digital Preservation tools

In the following section we are presenting three different approaches of testing in Digital Preservation. At first we show the DELOS testbed [19] which tests tools to find the most suitable preservation strategy for an individual use case. We present the Planets testbed [20] which is an online platform for testing certain preservation tools. Finally we show a project [7] by the National Library of Australia which tests file characterization and metadata extraction tools to investigate the capabilities and effectiveness of them.

The DELOS Testbed

There is a large variety of tools and strategies for Digital Preservation. The selection of the most suitable preservation approach for individual requirements is really complicated. In [19] a testbed which measures the performance against well-defined objectives is introduced. Due to predefined requirements it should facilitate the search for the right preservation action for a certain use case. Figure 2.1 provides an overview of the DELOS testbed workflow which consists of three phases and 14 steps.
At first the user has to define the scenario, choose the sample records and describe the requirements which should be fulfilled by the preservation solution. The sample records are taken from file repositories and needed to run the experiments. For the evaluation the user has to assign measurable units for each objective. These objectives can be categorized into four types:

- File characteristics
- Record characteristics
- Process characteristics
- Costs

In the second phase the user defines several alternatives of preservation solutions which should be evaluated against. He specifies the resources that are needed to run the evaluation experiments. Thereby the user estimates the amount of work, time and money. After the estimation of resources the user compares the resources with the requirements and decides whether a proposed alternative should be evaluated or not. If the user decides to evaluate a preservation solution he will develop the experiments. In order to run repeatable tests he documents the settings. Then the performance tests will be run with the defined sample records. These records don't contain any annotations or metadata. The tests don't consider the correctness and correct behavior of the preservation solutions.
In the last phase the measured values will be normalized for the comparison of results. The user defined a large number of objectives but not all of them are equally important. For this reason

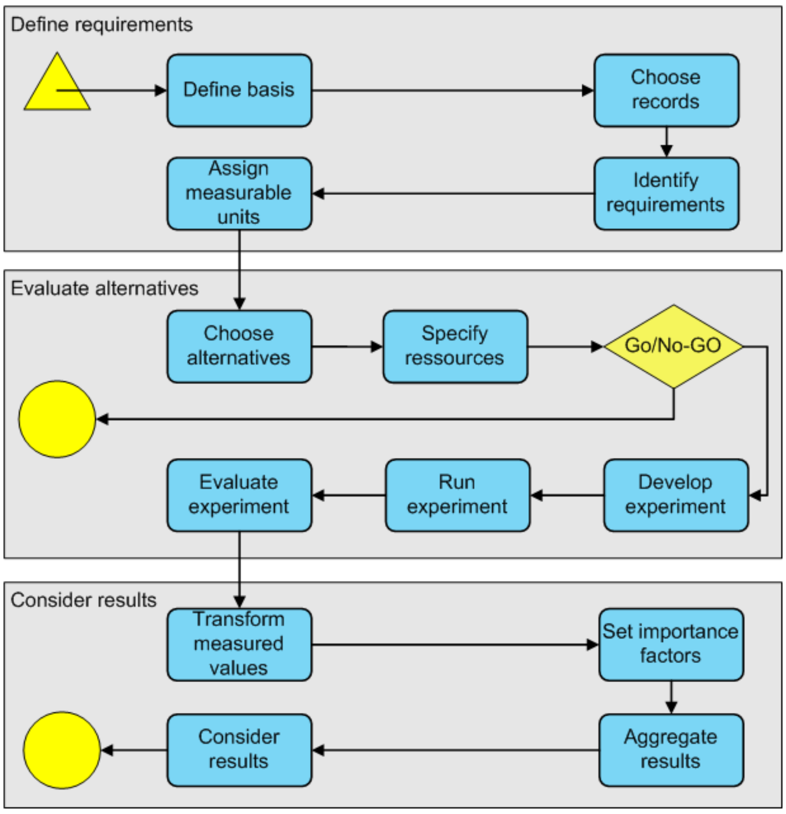


Figure 2.1: The DELOS testbed workflow [19]

the user has to set priorities which objective factors are important. A sensitive analysis which delivers a ranking of alternative solutions based on the measured values of the performance tests is done. Finally the user receives a ranking of preservation strategies and knows which solution fits best to the defined objectives.

The Planets Testbed

Testing and evaluating of characterization, migration and emulation tools in the field of digital preservation is supported by the Planets Testbed. In [21] [20] the Planets testbed is introduced which represents an environment to run experiments for several preservation tools. The test results can be empirically compared within the Planets Testbed. As figure 2.2 illustrates the core functionality of the Planets Testbed consists of three phases and six steps. The whole system is a web application. At first the user has to define the experiment. He must specify the basic properties which are important to find the results of an experiment on the Planets testbed system. In a second step the user designs the experiment thereby he selects one of the available preservation services of the testbed. The selected service will be tested. It needs input files which can be new files which are uploaded by the experimenter or existing files of the Planets data registry. Based on the type of the digital objects and the kind of the selected preservation service the user has to define the outcomes. Therefore the environment lists all the evaluation criteria and the experimenter has to select the desired. After these definitions the system automatically checks whether the experiment should be executed or not. If the experiment is refused by the system an administrator must check it manually.

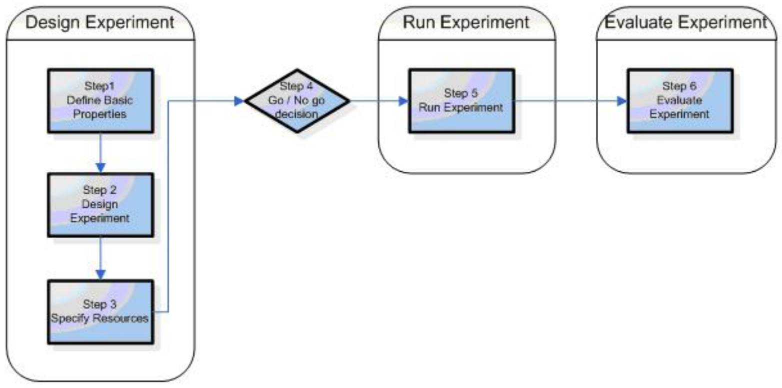


Figure 2.2: The Planets testbed workflow [21]

In a second phase the system will run the experiment and provide some performance measurements and statistics. The runtime of the experiments depends on the number and size of files.
Finally the user must define for each evaluation criterion comparison values for the input and the output files. Moreover he has to rate the evaluation of these criteria. The system analyzes the expected evaluation values with the actual values of the inputs and outputs, and posts the results to the Planets testbed. Other users of the testbed have the possibility to reuse the test results or give comments about them.
In [21] B. Aitken et al. describe what results are expected if a user tests one of the following three kinds of preservation tools:

- Characterization tool: The result of the testing should show that the characteristic tool characterizes the properties of a file correctly.
- Migration tool: The result should show that the properties of a file remain untouched.
- Emulation tool: The result should show that the emulated values are the same as in the original file.

Testing Preservation Tools at the National Library of Australia

In 2012 the National Library of Australia launched a project to investigate and test software tools for digital preservation of potential interest. They published their test results on file characteri-

zation and metadata extraction tools in [7]. They looked about the capabilities and effectiveness of diverse tools.
To run the experiments the National Library of Australia needed a collection of test files as input for the tools. As stated in [7] there is a big problem that there does not exist collected data with well-established references which the tool's results can be tested against. Therefore files without any annotations or metadata had been used.
For the tests the National Library of Australia developed a software framework which automated the testing workflow. The aim of the framework was to process a large number of files efficiently, to work robustly and predictably in the case of a processing error, to allow stops and restarts before the system is terminated and to produce results in a form which can be compared.
The project followed two goals on the one hand they wanted to investigate the reliability, correctness and usefulness of the results the tools produce. They came to the conclusion that without knowing the correct answers of certain test data, the correctness of a tool is not possible to measure. Therefore they compared the results of diverse tools and looked what they can learn from them. On the other hand they investigated and demonstrated approaches to process a large load of files and manage their results. The results are documented in [7].

2.2 Benchmark data

In [5], [6] and [7] it is stated that benchmark data is necessary to evaluate the results of preservation tools. In [6] R. Neumayer et al. give an overview of issues arising while building or developing such a benchmark data corpora for digital preservation. The observations and contents presented in [6] are the basis for this section.

2.2.1 Introduction

In general the evaluation of tools is done by comparison of results. For example in [7] M. Hutchins compares the test results of diverse tools and decides about the correct behavior of a tool. Benchmarking corpora is very important to test algorithms. Therefore a benchmark data corpus should contain annotated collections of files. These annotations are a manual classification of digital objects and represent the criteria the algorithms or tools will be evaluated against. Building a benchmark corpus is a non trivial task. It is important to take account what will be done with it. In general there exist the following two use cases for it:

- ***System quality evaluation:*** It should measure the correctness of algorithms or tools with respect to quality of their results.
- ***System performance evaluation:*** It should show the scalability of a system and the performance per instance or object.

To guarantee a high quality of benchmarks the benchmark corpora must satisfy certain quality constraints. In Digital Preservation the benchmark corpus should hold a huge number of digital objects and can be applied to a broad range of different institutional backgrounds [6]. As defined in [6] benchmark corpora can be categorized by their purpose:

- ***Content-complete corpus:*** It should cover *"the widest variety of possible types of content available in a given scenario. The purpose of such a corpus is the testing of organizational procedures: for instance, all the digital object types used in a given organization or settings are covered, in order to describe their specific problem definitions [6]."*

- ***Feature-complete corpus:*** It *"is defined by the coverage of the widest variety of possible features of a given object type. Such a corpus therefore is, by definition, object-type specific, and can be used to test the completeness of implementations for a given object type [6]."*

- ***Performance-defining corpus:*** It is *"a set of objects which is sufficiently large so that two or more programs processing it can be compared in a meaningful way with respect to their performances. A combination of both 'content-' and 'feature-complete' corpora would make a good choice for a performance-defining corpus [6]."*

2.2.2 Challenges for Digital Preservation Corpora

R. Neumayr et al. introduces in [6] the following five big challenges for corpus generation in digital preservation:

- ***Precise Task Definition*** is important for the usefulness and success of a corpus in digital preservation.

- ***Size of a corpus*** can be seen as the number of digital objects it holds or the corpus size in terms of hard disk space. The size of a corpus always depends on the concrete task. For example for measuring the scalability of a system a big corpus would be necessary. In contrast for quality measurement a small content complete corpus with a few files may be enough.

- ***Stratification*** is the coverage of types of digital objects, domains and varieties which are needed by the user community. For example this can be the distribution of elements according to different criteria. These stratification categories can be file type, time or real-world scenarios.

- ***Data Representation*** of the corpus should be in a machine readable and process able form. This is necessary because otherwise it wouldn't be possible to compare results.

- ***Ground Truth*** is the criteria where a result should be evaluated against. This evaluation can be done by humans or machines. The ground truth is necessary to judge the correctness of a system. Every digital object within a corpus should be annotated with a ground truth.

2.2.3 Existing Benchmark Data Collections

Several benchmark data collections exist. Most of them are used in Digital Forensics, Information Retrieval or Machine Learning. They are used to evaluate the performance or quality of a system. In the following section we present a few benchmark data collections of different types. For example we show text-, image-, video and audio benchmark data collections.

Reuters Corpora Volume 1 (RCV1)

The Reuters Corpus Volume [22] is an archive of Reuters news stories written in English. It holds about 810,000 newswire stories distributed in XML and plain text. The corpus is designed for research and development in the field of information retrieval or machine learning. In 2004 the National Institute of Science and Technology (NIST) took over the distribution of RCV1 and any future Reuters Corpora: Further Reuters Corpora are the Reuters Corpora Volume 2 [19] which holds 487,000 news stories written in thirteen languages and the Thomson Reuters Text Research Collection (TRC2). The TRC2 holds about 1,800,370 news stories and is available via web download.

Text Retrieval Conference(TREC)

The Text Retrieval Conference [20] [23] organizes yearly conferences to specific topics in the field of Information Retrieval (IR). It encourages research in IR through large text collections. Moreover they provide a lot of other kinds of document collections (e.g. Video Collection) for research in IR.

Learning to Rank (LETOR)

Ranking of information is the central problem of Information Retrieval. To compare existing learning algorithms benchmark data is necessary. Therefor the Learning to Rank (LETOR) [21] [24] project developed a benchmark data collection. They created two large datasets which *"contain queries, the contents of the retrieved documents and human judgments on the relevance of the documents with respect to the queries [24]."*

Dresden Image Database

The Dresden Image Database [25] holds 14,000 images which have been acquired under controlled and widely comparable conditions. They used 73 digital cameras to take photos and documented the settings of each. The corpus is built for development and benchmarking of camera based Digital Forensic technologies. It is freely available for scientific purposes and avoids copyright or privacy issues.

Kodak consumer video benchmark data set

The Kodak consumer video benchmark set [26] includes a few thousand videos from actual users who participated in a user study over one year. The videos are from a user generated content site (YouTube). The benchmark data set also includes a rich lexicon with more than 100 semantic

[19]http://trec.nist.gov/data/reuters/reuters.html

[20]http://trec.nist.gov/

[21]http://research.microsoft.com/en-us/um/beijing/projects/letor//

concepts and annotations over the video data set. It is used for semantic indexing of images and videos. It should facilitate the development and evaluation of large-scale learning based semantic indexing techniques [26].

Benchmark Dataset for Audio Classification

In [27] H. Homburg et al. present a freely available benchmark data set for audio classification and clustering. It should support developers to evaluate their audio classification and clustering algorithms. Moreover it gives researchers the opportunity to test the performance of their methods on heterogeneous data. The benchmark data itself contains about 1886 songs which hold metadata about their songs.

Music Retrieval Evaluation Exchange (MIREX

The Music Retrieval Evaluation Exchange [28] is a community-based formal evaluation framework coordinated by the International Music Information Retrieval System Evaluation Laboratory (IMIRSEL). They announce several tasks every year and within the community these tasks should be solved. These tasks can be seen as contests in developing techniques in the domain of Music Information Retrieval. For these contests they use a benchmark data set which is provided by the MIREX community.

2.2.4 Obeservations

A benchmark collection is defined for a special use case. The most common use cases are the system quality and performance evaluation. It is important that every object within a benchmark collection holds a ground truth for evaluation purposes.
This section introduced diverse benchmark data collections in the fields of Digital Forensics, Information Retrieval and Machine Learning. While the Reuters and TREC benchmark corpora focus on text materials we introduced benchmark collections with focus on multimedia contents. For example they hold videos, images and audio files. The introduced benchmark corpora also differ in their size and their field of application. But all of them support researchers with the evaluation of their investigated artifacts.
Digital Preservation suffers on a profound lack of annotated benchmark data collections. Within this thesis we show a way to fill this gap. All of the introduced benchmark collections are based on existing data sets. As stated in chapter one we will follow a new approach. Instead of analyzing existing data sets we generate them from scratch. For the generation of our benchmark data sets we are using the Model Driven Architecture which facilitates the usage of well-known model driven engineering tools and frameworks.

2.3 Model Driven Architecture (MDA)

The Model Driven Architecture is the underlying concept of our model driven benchmark data generation framework. In the following chapter which is based on [29], [30] and [31] we describe the concepts and components of it.

2.3.1 Introduction

In 2001 the OMG [22] (Object Management Group) adopted the Model Driven Architecture (MDA). MDA *"defines an approach to information systems specification that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform [32]."*
As stated in [30] the MDA follows the principle of Separation of Concerns which divides the process of software development into several abstraction levels. Every level is represented by a formal model which is a description or a specification of the system. To define such models modeling languages will be used.
With MDA it is possible to specify a system independently of the platform that supports it. MDA follows the goals of interoperability, portability and reusability [30].
Figure 2.3 illustrates the MDA software development process which is described in [29] and [31]:

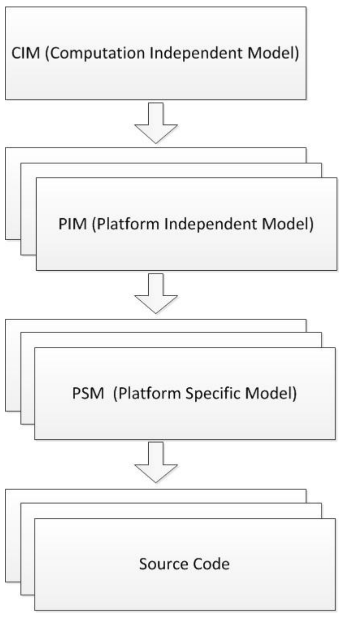

CIM (Computation Independent Model)
PIM (Platform Independent Model)
PSM (Platform Specific Model)
Source Code

Figure 2.3: Key elements of the Model Driven Architecture [29]

The Computation Independent Model (CIM) is the most abstract level and the business view of the solution. In the next step the Platform Independent Model (PIM)is defined. It is a description of the system. It contains information and algorithms independent from the implementation technology. For example, a PIM is a specification of the architecture of a system without technical details. Through transformation the PIM is transformed into a Platform Specific Model (PSM). The PSM is a technology-aware detailed specification of the system. In the last step the PSM is transformed to source code for a specific target platform.

[22]http://www.omg.org/

2.3.2 Software Development with the MDA

As described in [31] MDA is a model driven approach of software development where models are used to manage the development process. Models describe a software system and it‘s environment regarding a certain purpose. These models are defined in a formal language which is defined by syntax and semantics that a computer can interpret [30].

Models of the MDA

As mentioned in the previous chapter every model represents another abstraction level. The OMG [30] defined three different models to describe a software system: The CIM, the PIM which is a concrete representation of the CIM and the PSM which is a concrete representation of the PIM. This section is based on the definitions of [29].

Computation Independent Model (CIM): It is the most abstract model. It defines the *“context, requirements, and purpose without any binding to computational implications [29].”* For example it describes exactly the business processes, the design of a system, the involved persons or departments, the relations between the processes and so on. It hides all the staff in connection with the implementation platform of the system because it remains independent on how a system will be implemented. The CIM can be transformed to several PIMs. But it isn’t a necessary component of the MDA.

Platform Independent Model (PIM): It is at a very abstract level because it is independent of the implementation platform. It describes the behavior and structure of a subsystem of the CIM. The PIM knows the workflow and functionalities or interactions of the system. Moreover the PIM *“exhibits a sufficient degree of independence so as to enable its mapping to one or more concrete implementation platforms [29].”* Through transformation several PSMs can be generated from a PIM.

Platform Specific Model (PSM): It is the result of a transformation from a PIM. This model contains all the information about the structure and behavior of a system on a specific platform. The PSM can be seen as a pre stage to the source code of a system. Most of the systems use more than one technology therefore it is necessary to generate several PSMs from a PIM.

Source Code: The last step of the MDA development process is the transformation of the PSMs into Source Code. The source code is the lowest abstraction level of the MDA [31].

Transformations

The most important concept in MDA is the transformation of one model into another one or source code. In MDA there are two important kinds of transformations. On the one hand there is the model to model-transformation which is used to transform a PIM into several PSMs. On the other hand there is the model to text transformation which generates the source code out of a PSM. In this section we present them based on the explanations of [30], [31] and [29].

Model to Model Transformation: A Model to Model Transformation takes one or more models as input and produces one or more models as output. But it is sufficient that there is at least one input model and one output model. The transformation process itself is defined by transformation rules which define the mappings between the models. Moreover such a transformation can introduce additional information. For example it can insert some platform specific parameters
In MDA a model to model transformation is used to transform a PIM into a PSM. Figure 2.4 illustrates such a transformation:

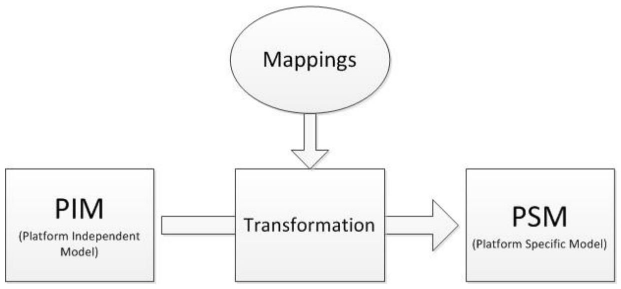


Figure 2.4: Model to Model Transformations [29]

Thereby out of one PIM several PSMs can be produced. To define such a transformation a mapping language is used, we explain such a language later.

Model to Text Transformations: A Model to Text Transformation uses a template-based approach to generate Text or Code. In general such a transformation gets a model and a corresponding template as inputs. In MDA a model to text transformation is used to transform a PSM to Source Code. Figure 2.5 illustrates such a transformation.

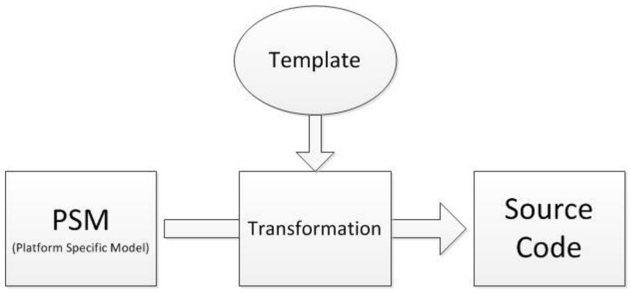


Figure 2.5: Model to Code/Text Transformations [29]

2.3.3 Evaluation of the MDA

In the following section we summarize the observations about advantages and disadvantages of applying the MDA which are stated in [33] and [34].
The MDA provides models on a very abstract level which allow the representation of complex

systems in a compact manner. For example a PIM can organize a complex software system regardless of the technical details. This simplifies the process of software engineering and these models are easy to understand. They can be used to enhance the understanding of the problem domain between a developer and customer.
The most common use case of MDA is the modeling of the structure of a system. The modeling of dynamic behavior is problematic. For the successful generation of the application, logic would be necessary. MDA tools provide functionalities for defining the behavior, which is more like a programming task and not a modeling task.
Computer science technologies underlie rapid changes. For example an actual programming language is replaced by a new version. MDA, however, permits a rapid adoption to new technologies. Therefore it is possible to derive a PSM corresponding to a new technology from the PIM.
Automated code generation with the MDA accelerates the development process rapidly. Thereby the stereotypically programming tasks will be replaced by re-use. Moreover this decreases the probability of programming mistakes. Due to the fact that the whole code generation is done automatically code optimizations are limited. For example it is hard to increase the performance of the generated code.
The MDA consists of a standardized modeling language which guarantees the interoperability between the models and tools. For this reason it is easy to realize the model transformations with diverse technologies.
Changing or editing artifacts of the MDA can lead to problems. It is really hard to keep changes consistent over the whole MDA process. For example a simple change in the PIM can require changes in the PIM to PSM transformation, the PSM and the Code generation. As described a simple change can cause a chain of changes.

2.4 Eclipse Modeling Framework

The Eclipse Modeling Framework (EMF) is an open source Java framework which supports model driven engineering. It exploits the Eclipse environment with technologies that facilitate the handling and working with models. The following section gives an introduction to the Eclipse Modeling Framework and its transformation technologies.

2.4.1 Introduction

F. Budinsky et al. gives in [35] a good overview of the Eclipse Modeling Framework and its capabilities. Their book forms the basis for this section. In EMF you can generate a model instance out of Java, XML or UML. You can also generate with one of them the others and the corresponding implementation classes [35]. Figure 2.6 *“shows how EMF unifies the three important technologies: Java, XML, and UML. Regardless of which one is used to define it, an EMF model is the common representation that ’glues’ them all together [35].”*

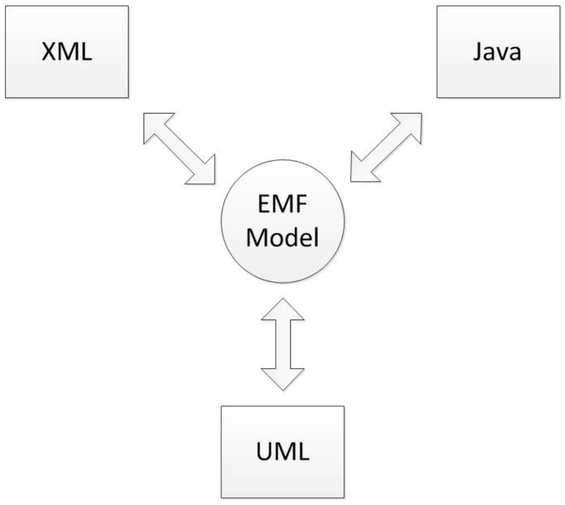


Figure 2.6: EMF overview [35]

For example you can define a structure of a bill in XML. Then you can automatically generate the UML diagram for the structure of the bill and in a further step you can generate Java code to manipulate the XML. This is one possible way to create an application. It is very convenient, because you must only define one of the three representations to generate an application.
The EMF Model is the central point of the Framework. It describes a lot of different things and must be handled by the framework. Therefor a common terminology to describe a model is necessary [35]. We need a model with information about the structure and elements of a concrete model instance. This kind of model is called a meta-model and describes an EMF model. In the Eclipse Modeling Framework the meta-model uses the Ecore model which is a meta meta-model for describing meta-models.

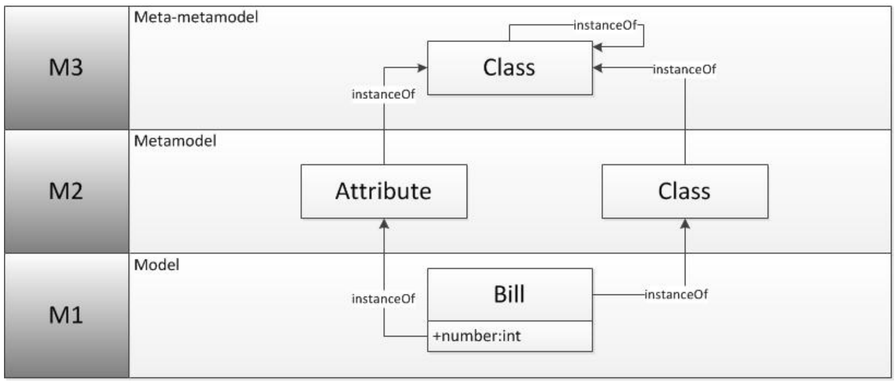


Figure 2.7: Meta modelling [29]

Figure 2.7 simplified illustrates the EMF modeling terminology. We have a concrete model which is like an UML diagram. It represents a class bill with an attribute number of type int. To describe this UML class a meta-model is used which defines the structure of the model. It defines an attribute and a class. These properties must be defined by a meta meta-model. It defines that every property of the meta model is an instance of class. Thereby the class is itself also an instance of class. The meta meta-model in EMF is called Ecore and is more complex than illustrated in figure 2.7.

2.4.2 Transformation between the models

As we discussed in the previous section of the chapter model to model transformations are the basis of the Model Driven Architecture. The Eclipse Modeling Framework provides several model to model transformation technologies. The Query/View/Transformation (QVT) and Atlas Transformation Language (ATL) are the most commons which are used within EMF. Both of them have a similar operational context which is shown in Figure 2.8.

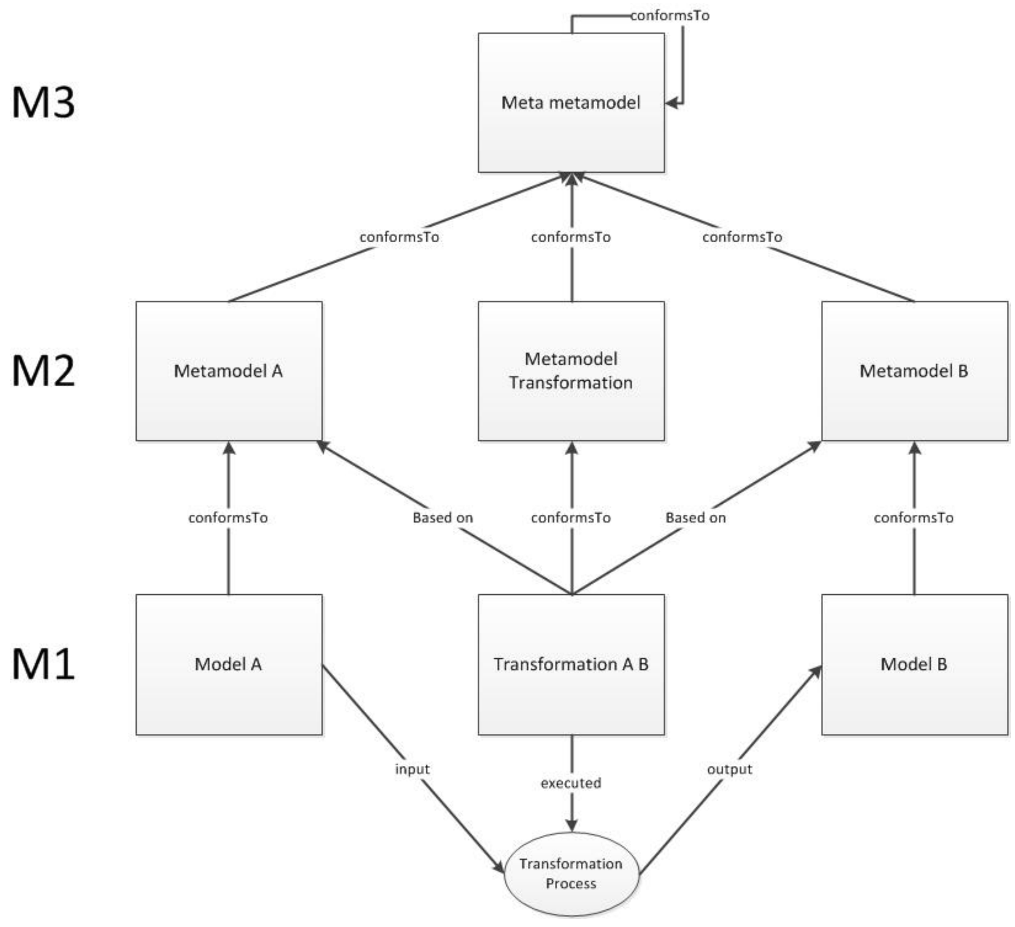


Figure 2.8: Operational context of QVT and ATL [36] [37]

As stated in [36] and [37], 'Transformation A B' is a program which generates 'Model B' out of 'Model A'. All three entities belong to metamodels. The 'Transformation A B' belongs to the 'Metamodel Transformation', the 'Model A' to 'Metamodel A' and 'Model B' to 'Metamodel B'. However, QVT and ATL define the 'Transformation A B' and have their own defined meta-models. All these meta-models conform to the same meta meta-model which is the highest layer in the transformation architecture.

Query/View/Transformation (QVT)

The QVT is a specification of the OMG which describes a language for model to model transformations [38]. The language delivers the following three functionalities [36]:

- ***Queries:*** They are used to select single elements of a model.
- ***Views:*** They are complex queries which give the possibility to select sections of a model.
- ***Transformations:*** Most of the time this functionality is used. It transforms one model in another model.

Moreover QVT enables the usage of external code or transformation languages to extend it [36]. In general QVT provides imperative and declarative programming. It is a really powerful language because it delivers in addition to the transformation functionalities other functionalities.

Atlas Transformation Language (ATL)

Jounault et al. give in [37] an introduction to the ATL transformation language. They state that ATL is a transformation language for automatic model to model transformations. It provides declarative and imperative programming, but in general the ATL consist out of a set of rules which define the mapping of elements between two models. The language only provides transformation functionalities but is seen as a *"QVT-like transformation language [37]."*

2.4.3 Code Generation

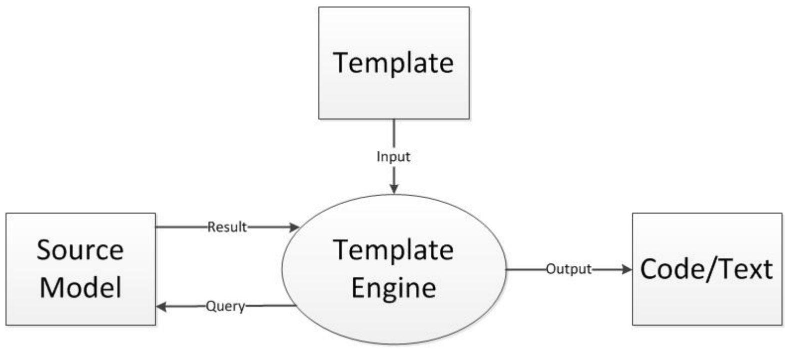


Figure 2.9: Overview Code Generation [29]

The last step of the Model Driven Architecture is the code generation. It uses a template based transformation language which separates dynamic and static code [29]. *"A template can be seen as a kind of blueprint which defines static text elements shared by all artifacts as well as dynamic parts which have to be filled with information specific to each particular case [29]."* Figure 2.9 illustrates a template based code generation approach which takes as input a template and produces text or code. For the dynamic contents the template engine has to query the source model. It returns these contents to the template engine and replaces the dynamic parts of the template with them.
M. Brambilla et al. introduced in [29] the following template based transformation languages:

- **XSLT:** It is the W3C standard for transforming XML files into text documents. Using XSLT implies that the programmer has the knowledge on how models are encoded in XML. For this reason technologies operating on model level are easier to use than XSLT.
- **JET:** The Java Emitter Template was one of the first technologies which generated text out of models. It can be also used to generate code out of 'normal' java based objects. JET doesn't provide any possibilities for querying models.
- **XPAND:** It is a template based transformation language for code generation. It operates on the model level and provides a dedicated language for querying models.

2.5 Webpage adaption

Today there is a large variety of different clients which are accessing the internet. These clients are mobile devices or different browsers. There is the need that the web contents are accessible with all of them. For example accessing a webpage with a browser is different from accessing it by a mobile phone. The mobile device has a smaller display and the contents should be illustrated in a readable and understandable form.
For this reason adapting of web content is necessary. Web content adaption can be defined as *"a process of selection, generation, or modification that produces one or more perceivable units in response to a requested uniform resource identifier in a given delivery context [39]."* A webpage can be seen as some kind of generic content which must be transformed in a platform specific representation. These transformations deliver us an optimized version of the web content for a specific device or browser.
Most of the adaption strategies use the DAD (Detection Deliver Text) mode described by Covic et al. [40]. This mode includes the following steps:

1. **Detection:** At first a client must be detected. Therefore the client must send information about itself to the content provider. For example the client notifies its type (e.g. smartphone).
2. **Adaption:** The adaption method transforms the contents based on the information gained by the detection
3. **Delivery:** The adapted content is delivered to the client.

2.5.1 Levels of adaption

Adzic et al. define in [41] three levels were adaptions are done.

Server-side Adaption

This adaption approach is done on a server. It is necessary to store the content for delivery on the server. These contents are modified through software or technology installed on the server. The main advantage of this approach is that the server can modify the contents for the specific needs of each client. For example the Extensible Stylesheet Language(XSL) is used to transform HTML webpages in an adequate representations for the client [42].
As described in [42], authors have the possibility to decide between the following three authoring methods:

- **Multiple Authoring:** They create different versions of the content for diverse user groups.
- **Single Authoring:** They provide one version of the content which will be transformed into a proper format for the client.
- **Flexible Authoring:** It is a authoring method which combines multiple authoring and single authoring.

Intermediate Adaption

This adoption methodology is performed between the server and the client. In [40] Covic et al. specify this kind of adaption as proxy server adaption. Covic et al. state that a proxy server is interposed between the web content provider (server) and the content consumer (client). The main advantage of this approach is that there isn't any technology for the content adaption on the server and the client.

Client-side Adaption

The third adaption method is the client-side adoption. The whole adaption process is done on the client. A client uses style sheets to format the web contents [40]. For example the usage of Cascading Style Sheets (CSS) constitute such an adaption approach. The main advantage of this approach is that it avoids the client communication to the server or proxy.

2.5.2 Examples for Content Adaption

As mentioned above the server side adoptions are done with XSL and the client side adaptions are done by Cascading Style Sheets (CSS). The intermediate adaptions admit the usage of a large variety of different approaches.
He et al describe in [43] an approach for flexible content adaption using a rule based approach. They introduce a system which is running on a proxy. The system is called Xadaptor and has its focus on adaption of web contents for mobile devices. He et al. use a rule based approach *tto facilitate extensible, systematic and adaptive content adoption [43].*" The system holds a rule

base which contains rules for various content types. The rules are invoked based on the information gained by the detection. The rules are written in Prolog and adapt the structure, content and pointer objects. While existing techniques focus on content objects, He et al. also deal with structure and pointer objects.
Another approach of intermediate adoption is explained by Lopez et al. in [44]. Lope et al. introduced a proxy server which takes a HTML page as input and transforms it with the Extensible Stylesheet Language (XSL) to an optimized webpage for small portable devices. Therefore several XSL transformations are defined to provide transformations for a lot of different devices.
Yang et al. introduce in [45] an intermediate adaption system. The system contains a decision engine which decides how the contents should be modified. The engine uses the client's capabilities to decide about the adaptions. Moreover the client can send adoption preferences to the system which are considered by the system. The main focus of the system is to provide an adaptive content delivery algorithm.
Laako et al. show in [42] an approach for web content adaption for mobile devices. In [42] a Extensible Hypertext Markup Language (XHTML) webpage is used as input for the proxy. This webpage is transformed in a structural tree (DOM tree). This tree is traversed by predefined rules. The rules edit, delete or insert nodes. Moreover the system can divide a tree in several sub trees. This makes it possible that one webpage can be transformed to smaller webpages. These subpages are linked together with hyperlinks. The whole system is Java based and generates XHTML mobile Profile and Wireless Markup Language (WML) webpages.

2.5.3 Observations

As mentioned above webpage adaption is the process of transforming generic webpages to platform specific representations to fulfill the client's needs. This issue stands in connection with our topic of generating webpages as benchmark corpora. Due to the fact that we use the Model Driven Architecture methodology, we also transform platform independent webpages representation into platform specific ones. For this reason we did a closer look on the projects and systems dealing with webpage adaption. The adaptions are done with Extensible Stylesheet Language, Prolog programs, DOM tree manipulation, Java programs and decision engines. But all of them are using a set of rules which define the adaptions which should be performed. All in all the webpage adaption showed us that the generation of platform specific representations is commonly used in practice.

2.6 Summary

In this chapter, we gave an introduction to the topics on which this works is based. We introduced the Digital Preservation, its challenges, the current state of art and activities which are used to guarantee error free long term storage of information.
We then introduced the issue of testing and finding the most suitable digital preservation action for a particular use case.
We outlined the problem of missing annotated benchmark corpora in digital preservation and for what a benchmark corpora is necessary. Then we presented existing benchmark data collections

in the field of Digital Forensics, Information Retrieval and Machine Learning. We also showed the challenges which have to be faced for creating an annotated benchmark data collection.
Then we introduced the Model Driven Architecture which is used in this work. We explained the model to model transformation and code generation which is necessary for applying the architecture.
We introduced the Eclipse Modeling Framework which is used to apply the Model Driven Architecture. We outlined the main concepts and technologies of the framework.
Finally, we did a look on webpage adaption which deals with the topic of generating platform specific representations of webpages.

CHAPTER 3

A Model Driven Benchmark Data Generation Framework

The following chapter introduces the Model Driven Benchmark Data Generation Framework. At first we give an introduction to the backgrounds of the system. In a further step we provide an overview of the system. We explain all the involved models and processes in detail. Finally, we give an overview of the applied framework, model to model transformation and code generation technology.

3.1 Introduction

As stated in chapter 1 there is a profound lack of annotated benchmark data in Digital Preservation. Benchmark data with corresponding ground truth is necessary for the evaluation of new approaches.
As described in chapter 2 a few projects (Planets, DELOS, National Library of Australia) deal with testing. For example the National Library of Australia outlines in [7] that there is a big problem with the evaluation of preservation tools because of the inexistence of annotated benchmark data collections. They came to the conclusion that without knowing the correct answer of certain test data, the correctness of a tool is not possible to measure. Also C. Becker et al. state in [5] that "*annotated benchmark data are needed to support the objective comparison of new approaches and quantify the improvements over existing techniques [5].*"
In chapter 2 we did a closer look on existing benchmark data collections in computer science. We found out that there exist a lot of diverse benchmark data collections in the fields of Digital Forensic, Information Retrieval and Machine Learning. All of them are based on data collections which have been analyzed and annotated with a ground truth. For example the Reuters Volume Corpora provides Reuter's news stories as benchmark data collections.
The challenge of this work is to find, implement and evaluate a new approach to generate benchmark data collections with corresponding ground truth. The ground truth is the most significant

point of our benchmark data collection because it should deliver values and metrics to evaluate against.
Our benchmark data generation framework should fulfill the following criteria:

- It should generate benchmark data collections with corresponding ground truth.
- The generated ground truth should support the traceability of features, describe the structure and properties of a webpage.
- Instead of analyzing existing data collections our system should generate the benchmark data collections from scratch. Thereby our system should use statistically distributions of webpage features.
- It should solve the challenges of benchmark corpora defined in 2. (e.g. Size of Corpus, Stratification, Data Representation, Precise Task Definition, Ground Truth)
- It should not rely on proprietary formats and should use well-known technologies.
- It should work as automated as possible.
- It should be scalable and should produce a large amount of annotated webpages in an adequate period of time.
- It should be extendable.
- It should be easy to install and use.

Our system is based on the Model Driven Architecture which is described in chapter 2. We used this software development methodology to avoid platform and proprietary format dependencies. Due to the fact that the Model Driven Architecture is standardized, it is compatible with well-known model driven engineering frameworks and technologies. We are able to generate information objects which are 'natively born' in common environments.

3.2 Overview of the workflow

As stated in the introduction our Model Driven Benchmark Data Generation Framework is based on the Model Driven Architecture. Figure 3.1 gives an overview of the workflow of the system. The rectangles in figure 3.1 represent models. The ovals illustrate transformation processes and the arrows define the directions of the workflow.
As illustrated in figure 3.1 the workflow consists of four processes:

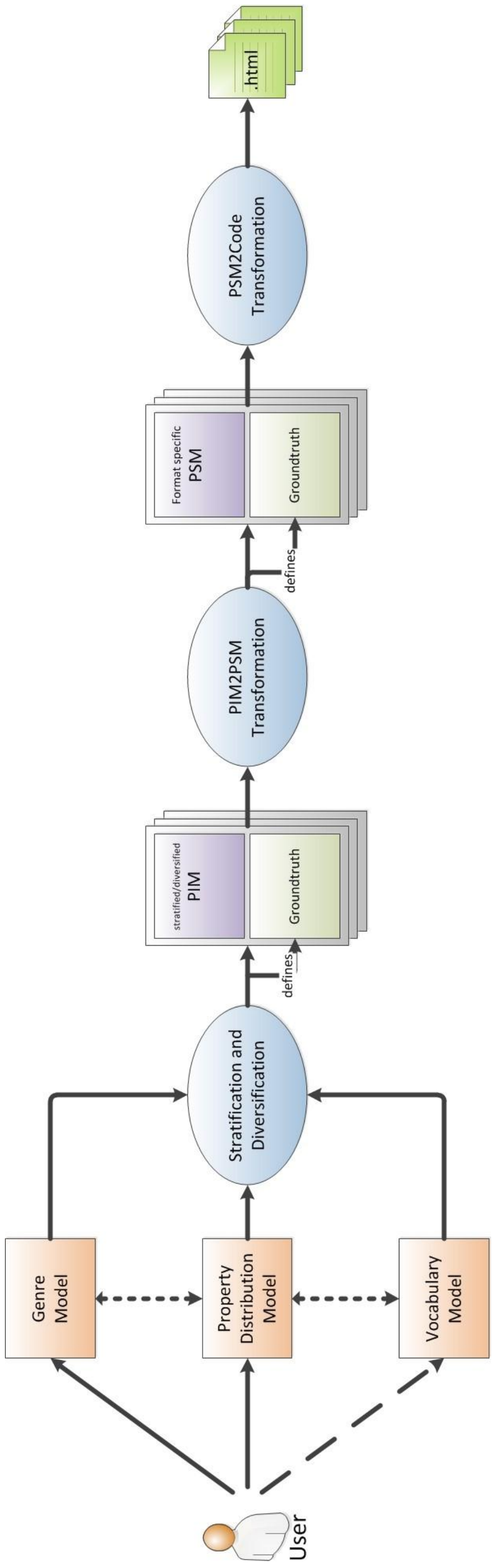


Figure 3.1: Workflow of the Model Driven Benchmark Data Generation Framework

1. **Initial user definitions:** The following model must be defined before running the framework:

    a) ***Property Distribution model:*** It must be defined by the user and holds the statistically distribution of features of a certain amount of webpages. For example the user defines that ten pages should be generated, where 30 percent of them contain two pictures and 70 percent have one picture.

    b) ***Genre model:*** It must be also defined by the user and specifies the layout of the generated webpages.

    c) ***Vocabulary model:*** The system holds a predefined vocabulary model. It defines the content which will be randomly inserted to the generated webpages. It is a data store. For example it holds picture, videos, paragraphs, headings, hyperlinks and so on. The user has the possibility to edit or extend this model.

2. **Stratification and diversification process:** After the initial user definitions the system can be started. The first process of the workflow is the stratification and diversification. Thereby for every webpage a PIM with corresponding ground truth is generated. The PIMs are based on the models of the initial user definitions process.

3. **PIM to PSM transformation process:** As defined by the Model Driven Architecture the PIMs are transformed to PSMs. This process also generates a ground truth for every PSM.

4. **PSM to Code transformation process:** Finally the system transforms the PSMs to HTML4 Code. It is the result of the Model Driven Benchmark Data Generation Framework.

3.3 Models of the system

In this section we explain all the models which are part of the Model Driven Benchmark Data Generation Framework.

3.3.1 Property Distribution Model

The Property Distribution is one of the most important components of the workflow to receive more practical webpages. We rely on Shannon's theory of entropy of printed English which is described in [46]. In this paper he investigated "the discrete source of information" and pursued the question how an information source can be described mathematically. He stated that *"In telegraphy, for example, the messages to be transmitted consist of sequence of letters. These sequences, however, are not completely random. In general, they form sentences and have a statistical structure of, say, English. The letter E occurs more frequently than Q, the sequence TH more frequently than XP, etc [46].“* Moreover Shannon explained the following about a discrete information source: *"We can think of a discrete source as generating the message, symbol by symbol. It will choose successive symbols according to certain probabilities depending, in general, on preceding choices as well as the particular symbols question. A physical system, or*

mathematical model of a system which produces such a sequence of symbols governed by a set of probabilities, is known as a stochastic process [46].“ To sum up, Shannon stated that there exists a statistical distribution how often a letter appears within a message.
We mapped Shannon’s theory of entropy of letters to our challenge of generating more realistic webpages. Instead of messages we have webpages and instead of letters we have features of webpages. Thus, we assume that every webpage has a distribution how often a feature appears. For example, features are pictures, videos, paragraphs, hyperlinks, headings, listings and tables. Based on these distributions we are generating our platform independent models and in consequence the webpages.
Moreover the concept with the distributions facilitates the usage of real-world distributions. We can incorporate them into our generation process. Therefore we need statistically data about webpage structures and contents. At the moment we are using manually defined frequency distributions of properties, therefore a program for converting real-world property distributions to our distribution model should be part of future research.

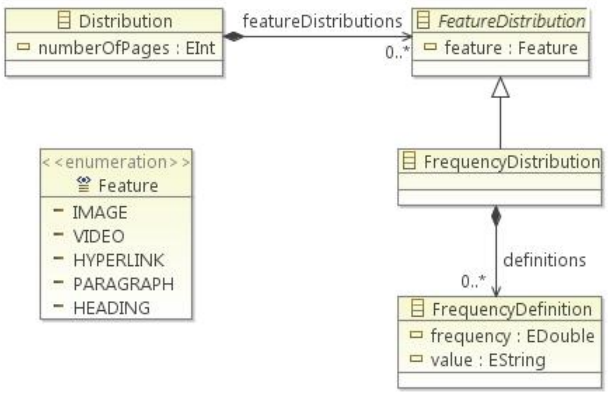


Figure 3.2: Property Distribution meta-model

Figure 3.2 illustrates the property distribution meta-model. The user must instantiate this meta-model to get a property distibution model. At first the user must define the number of pages that should be generated. This is also the amount of webpages for which the feature distributions are valid. A property distribution model holds a list of feature distributions. In the next step the user can define these distributions for the following features:

- Image
- Video
- Hyperlink
- Paragraph
- Heading

All these feature distributions are frequency distributions. Every frequency distribution consists of several frequency definitions. A frequency definition consists of a ’frequency-value’ pair.
For example we define an ’image’-distribution where 30 percent of the pages contain two images and 70 percent of the pages contain one. Moreover the total number of pages is ten. Table 3.1 demonstrates how the sample property distribution looks like. You can see that we selected ’Image’ as the kind of feature distribution and we have two ’frequency-value’ pairs. The user has

Frequency Distribution: ***Image***	
Frequency	***Value***
30	2
70	1

Table 3.1: Sample Property Distribution

the possibility to define more than one feature distribution within a property distribution model. After defining the property distribution model it would be used as input for the stratification and diversification process. The process will generate the defined number of pages with the corresponding feature distributions.

3.3.2 Genre Model

The genre model follows the goal of generating more practical webpages. It is used to define the overall structure and organize the content area of a webpage. For the layout or genre type there are two possibilities:

- **Blog:** Figure 3.3 shows the layout of a blog. It consists of two columns side by side. The left column contains a small picture and subjacent there are a few hyperlinks. The right column contains the content area of the blog.

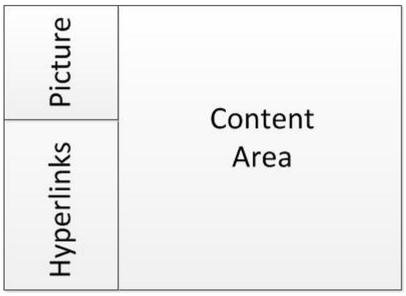


Figure 3.3: Genre Type Blog

- **Simple Page:** Figure 3.4 shows the layout of a simple page. It is based on two rows. The first row contains hyperlinks and the second row is the content area.

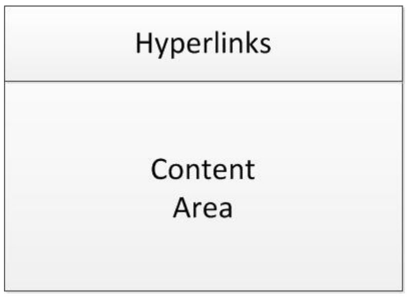


Figure 3.4: Genre Type Simple Page

Figure 3.5 illustrates the genre metamodel. Before running the framework the user must instantiate the genre model. He must select one of the two genre types. Then he can organize the content area by defining content patterns. A genre can have several content patterns which will be applied consecutively. Every content pattern has a number which indicates how often it should be repeated until the next content pattern will be used. The content pattern holds a list of content entities. A content entity consists of an index and a string which defines the type of entity.

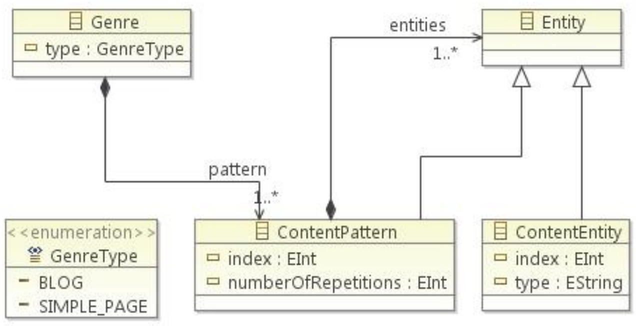


Figure 3.5: Genre meta-model

The following entity (type) strings are possible:

- ’Image’
- ’Video’
- ’Heading’
- ’Paragraph’

These four types are only abstract descriptions and don’t represent any content. The diversification and stratification process will replace all of them by ’real’ content with respect to the feature distributions. For example we define that our content area should hold a heading at first and then some text. This pattern should be repeated two times. Therefore we must instantiate a content pattern with number of repetitions two and we must define two content entities for it. The first content entity has the index one and the type 'heading'. The second one has the index two and type 'paragraph'.

Evaluation of the Genre model

The primary goal of the genre model is the generation of more practical webpages. For every genre type a template is predefined and is used at the stratification and diversification process.
The genre model defines the overall structure of the generated webpages and content area. It also enables the selection between different genre types. The advantage of the usage of the genre model is that we receive more realisitc webpages.
A big disadvantage of the genre model is that it stands in contact with the distribution model. It should cover the same features as the distribution model. If we change the distribution model then it is necessary to change the genre model. The same applies in the other direction. If we change the genre model then we must change the distribution model. For this reason we have a dependency between the property distribution and genre model.

3.3.3 Vocabulary Model

We introduced the distribution model and the genre model. These models help us to generate realistic webpages. The vocabulary model delivers us the contents for the webpages. It is a data store for the Model Driven Benchmark Data Generation Framework. It provides contents as videos, images, paragraphs, headings and hyperlinks.
Figure 3.6 illustrates the vocabulary meta-model which consists of the following three main content entities:

- **Video:** It links to the location on the internet where the video is stored (e.g. YouTube). The video entity contains the description, size and title.

- **Image:** It links to the location on the internet or file system, where the image is stored. The image entity contains the description, size and title.

- **Text:** In general every text entity holds its value, the value of the color and the size of characters. In our vocabulary model there exist the following three kinds of text entities:

  - ***Paragraph:*** It represents a long text which can be used as a paragraph.
  - ***Heading:*** It represents a short text which can be used as a heading. It holds information about the heding level. For example heading level one means that it is a subheading.
  - ***Hyperlink:*** It reperents a hyperlink which holds the location to link.

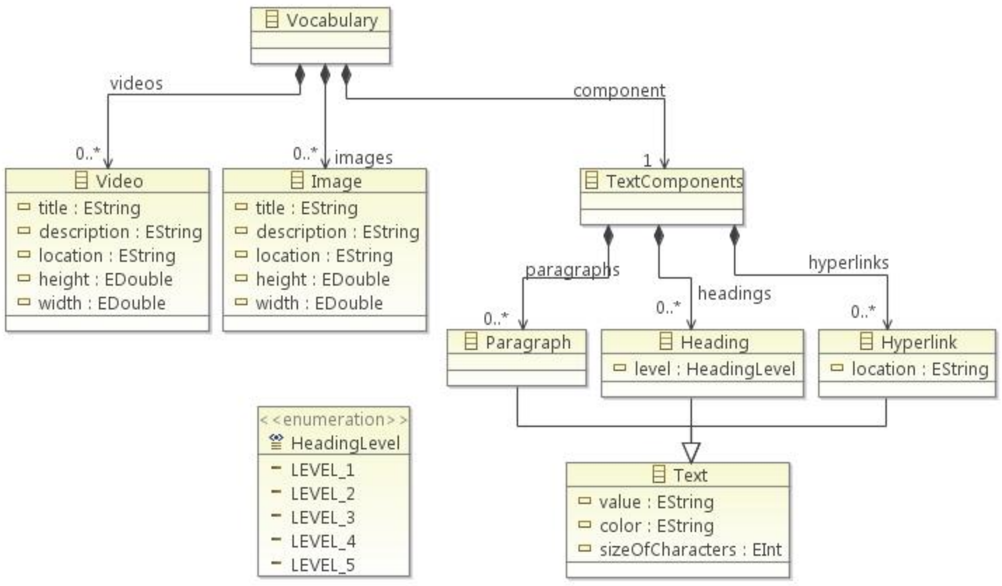


Figure 3.6: Vocabulary meta-model

Our framework contains a predefined vocabulary model which holds a few content entities of every type. It is possible to run the stratification and diversification process without defining further content entities. The user has the possibility to edit or extend the existing vocabulary model. For example he can increase the vocabulary with further entity definitions.

Evaluation of the vocabulary model

The vocabulary model is used to provide a data store for the diversification and stratification process. It delivers contents for the features. It stores contents for images, videos, paragraphs, headings and hyperlinks. This strict separation between the text components, especially paragraphs and headings is necessary to obtain more realistic webpages.
The advantage of the vocabulary model is the precise definition of the vocabulary. The system itself has a predefined vocabulary model which the user can manually edit or enrich with further data. It would be also possible to write an application which inserts automatically vocabulary to the model. This fact shows that the vocabulary model provides an interface for inserting contents to the system.
The disadvantage of the vocabulry model is the dependency from the distribution model. For the vocabulary model the same applies as for the genre model. If there is a change in the distribution model the vocabulary model must be changed and vice versa.

3.3.4 Platform Independent Model (PIM)

We defined a PIM for representing our webpages. Picture 3.7 illustrates the meta-model of the PIM which will be explained in this section.

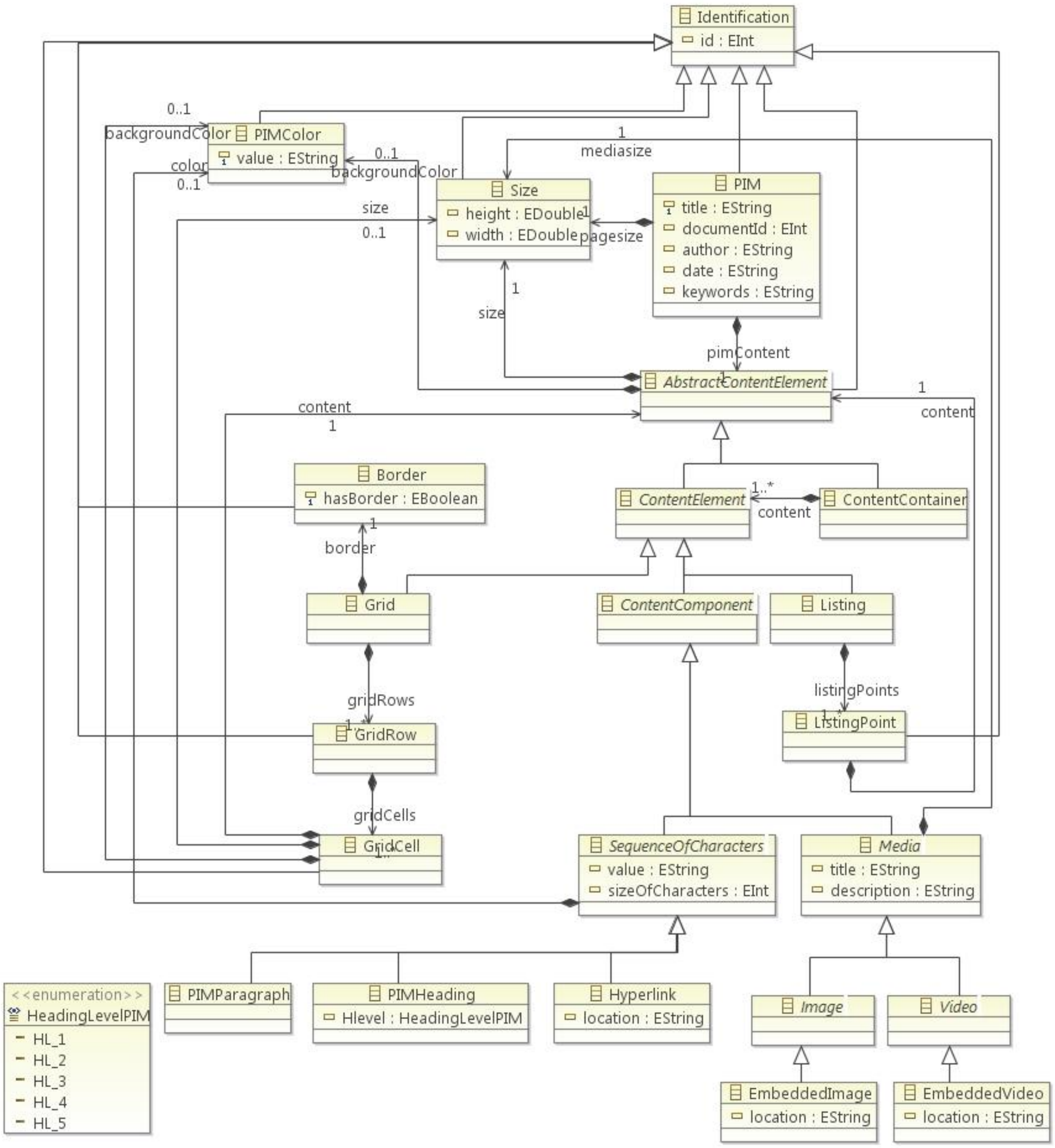


Figure 3.7: Platform Independent Model meta-model

Explanation of the PIM

For every webpage we instantiate a PIM entity. As illustrated in figure 3.7 it holds a title, author, date (of creation), defined size and unique document ID. This ID is necessary to associate other artifacts (e.g. ground truth) with a PIM. It is possible to tag the document with keywords which will be used to describe the contained information.
Figure 3.7 shows that a PIM holds at least one abstract content element which holds information about its size and the background color. As shown in figure 3.7 an abstract content element is the superclass of a simple content element or an array of content elements which is called content container. The content container is used if it is necessary to arrange diverse content entities after each other.
As shown by figure 3.7 the following three entities are subclasses of a content element:

- The ***grid*** entity holds information about the grid's border. It defines whether the grid has a border and the intensity of the border. The grid holds an array which contains grid rows. A Grid row contains several grid cells. Every grid cell contains information about the background color and its size. Moreover it contains an abstract content element.

- The ***listing*** entity contains an array of listing points. A Listing point contains an abstract content element.

- The ***content component*** entity is the superclass of the following two entities:
  - The ***media*** entity is abstract and holds a title and description. It can be either an image or a video. To keep the PIM as simple as possible we decided that we use embedded images and embedded videos. Both of them contain a String with the location of the image or video. Moreover they are subclasses from media.
  - The ***sequence of characters*** entity contains its value and the size and color of characters. This entity is an abstract superclass of the following four entities:
    - *Paragraph:* It represents a simple paragraph on a webpage.
    - *Heading:* It represents a heading on a webpage and holds information about the heading level.
    - *Hyperlink:* It represents a text which links to a webpage. Thus, it must hold the location of the webpage to link.

As illustrated in figure 3.7 the color entity defines the color of characters of a sequence of characters, the background colors of grid cells and abstract content elements. It holds a value which represents the color name.
To define the size of an element in the PIM a size entity can be used. It defines the height and the width. It is used to define the size of a document, abstract content element, media or grid cell.
The identification entity contains an id which is used for the ground truth. As described in figure 3.7 all the entities in the PIM inherit from it and have an unique Id.
Moreover this illustration shows the recursions which are necessary to interleave grids and listings, or to fill them with content components. The grid cell and the lising point entity realize this

recursuion because they hold a abstract content element entity. As mentioned above a abstract content element entity can be a content element or content container entity. The content element entity can be a grid, listing or content component. For example these recursions are used to fill a table with contents. A table cell can hold a further table, a listing or 'real' content (e.g. image, video, text).

3.3.5 Platform Specific Model (PSM)

The PSM can be defined for a lot of diverse platforms. This fact shows the power of the Model Driven Architecture approach. It is possible to map one PIM to a lot of different PSMs. For example it is possible to create a platform specific model for every kind of hypertext markup language (HTML).
Our PSM is for generating HTML4 webpages. Our formal way of creating the PSM was a detailed study of the HTML4 Specification [1]. Based on this study we developed a PSM which covers all the features of HTML4 except the HTML functionalities for generating forms. We did not include these features, because the generation of webpages with forms would not be necessary for our benchmark data generation system. Figure 3.8 illustrates the resulting meta-model which represents the HTML4 specification 1 without form features.

Explanation of the PSM

We define for every PIM instance a corresponding PSM instance. The Transformation between the PIM and the PSM will be explained in chapter 3.4.2. As stated before the PSMs are based on the HTML4 Specification which is explained in 1.

[1] http://www.w3.org/TR/html401/

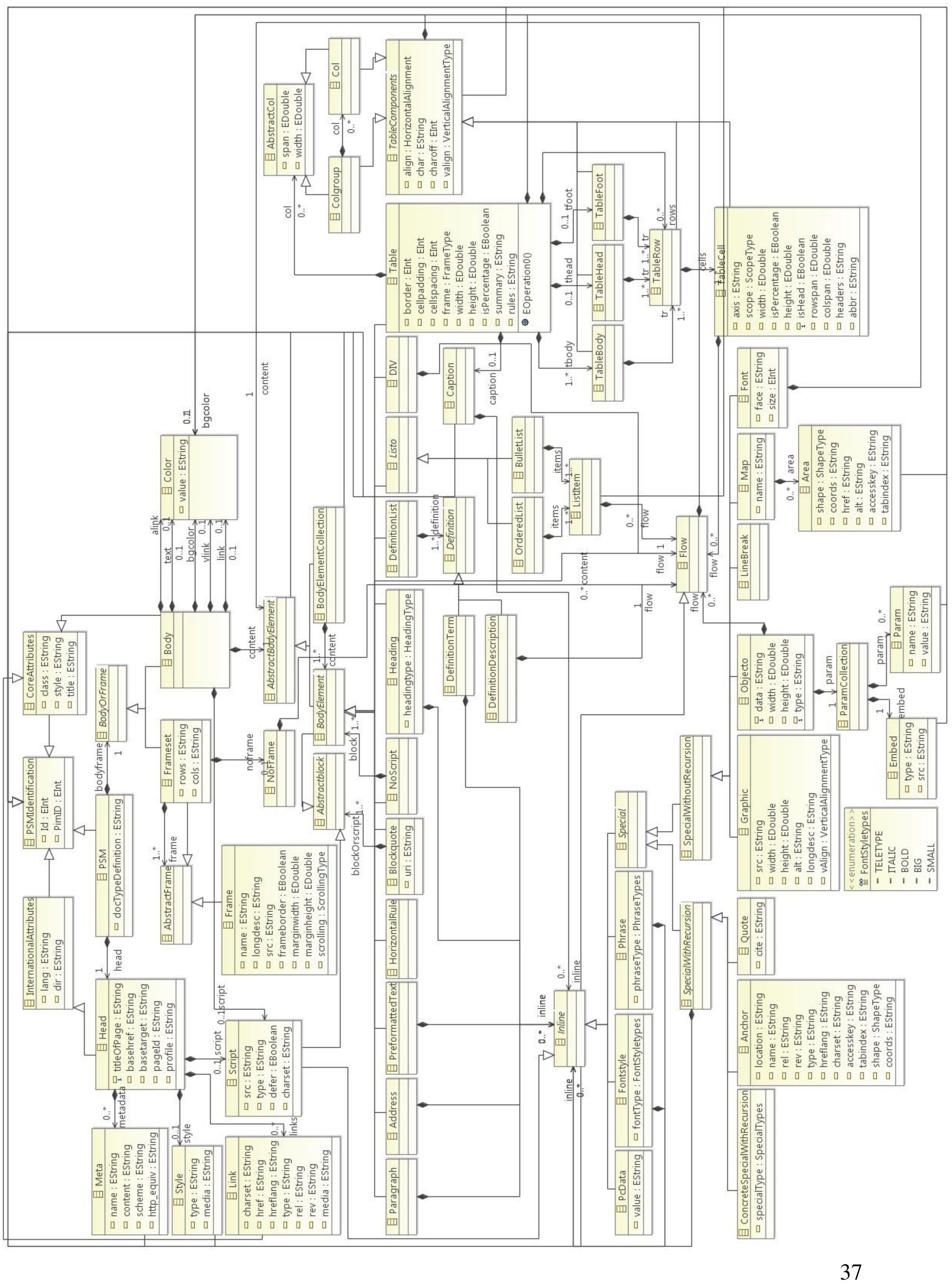
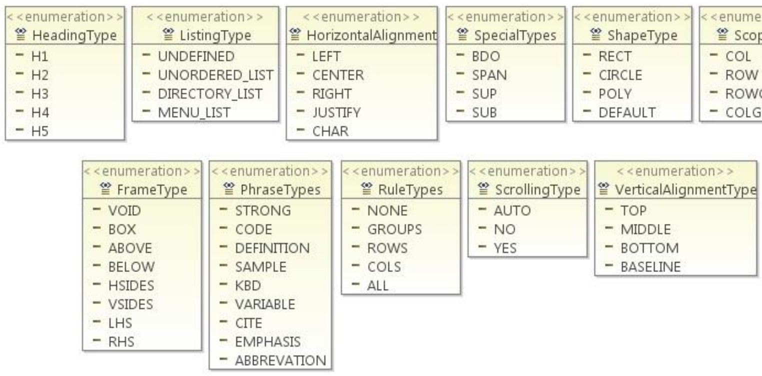


Figure 3.8: Platform Specific Model meta-model



<<enumeration>> HeadingType
H1
H2
H3
H4
H5
<<enumeration>> ListingType
UNDEFINED
UNORDERED_LIST
DIRECTORY_LIST
MENU_LIST
<<enumeration>> HorizontalAlignment
LEFT
CENTER
RIGHT
JUSTIFY
CHAR
<<enumeration>> SpecialTypes
BDO
SPAN
SUP
SUB
<<enumeration>> ShapeType
RECT
CIRCLE
POLY
DEFAULT
<<enumeration>> ScopeType
COL
ROW
ROWGROUP
COLGROUP
<<enumeration>> FrameType
VOID
BOX
ABOVE
BELOW
HSIDES
VSIDES
LHS
RHS
<<enumeration>> PhraseTypes
STRONG
CODE
DEFINITION
SAMPLE
KBD
VARIABLE
CITE
EMPHASIS
ABBREVATION
<<enumeration>> RuleTypes
NONE
GROUPS
ROWS
COLS
ALL
<<enumeration>> ScrollingType
AUTO
NO
YES
<<enumeration>> VerticalAlignmentType
TOP
MIDDLE
BOTTOM
BASELINE

Figure 3.9: Enumerators of the Platform Specific Model meta-model

In the following section we are explaining the PSM illustrated in figure 3.8 and 3.9. Details about the features can be found in the HTML4 specification. Figure 3.9 illustrates the enumerators which are used in the PSM.
As illustrated in figure 3.8 our PSM holds a head and body/frameset entity. The head entity includes information as title, address basis, links, style and metadata. For holding the contents of our webpages we are using a body or a frameset entity. This differentiation is shown in figure 3.8 through the abstract body or frame entity.
As illustrated in figure 3.8 the body entity holds an abstract body element. An abstract body element can be a body element or a collection of body elements. The collection is used to arrange body elements after each other.
The PSM illustrated in figure 3.8 holds the following body entities:

- **Paragraph:** It defines a simple paragraph on a webpage.
- **Address:** It provides contact information for a document or a section of the page.
- **Preformatted Text:** It contains preformatted text which can't be changed by the browser.
- **Horizontal Rule:** It defines a horizontal line.
- **Blockquote:** It defines a citation in an own content block.
- **No Script:** It offers an alternative content for a client-side script which won't be executed.
- **Heading:** It defines a normal heading and the heading level can be defined.
- **Definition list:** It is a list which is used to list several definitions for a term or topic.
- **List:** It is used to represent a listing. It differentiates between a bullet list and an ordered list. It holds listitems.

- **DIV:** It is a general container which can hold information about style and language of the container contents.
- **Table:** It represents a table. It holds table rows which contain table cell entities.

As illustrated in figure 3.8, the list item (part of the list entity), DIV and table cell entity (part of the table entity) hold a flow entity. The flow entity enables through a recursion back to the abstract body element the interleaving of body elements. A flow entity can be also an inline entity which holds concrete contents.
As illustrated in figure 3.8, the Paragraph, Address, Preformatted Text, Heading and Definition List entities hold the following inline entities:

- **PCData:** It is the value of an entity. (e.g. the text string of an paragraph)
- **Fontstyle:** It defines the style of the font (e.g. bold, italic or big)
- **Phrase:** It defines what kind of phrase it is. (e.g. definition)
- **Special:** It is an abstract entity and can be a:
  - ***Special entity without recursion:*** The following elements are special entities without recursion:
    - *Graphic:* It is used to embed an image.
    - *Object:* It is used to embed an object. For example an object can be a video.
    - *Line Break:* It is a simple linebreak.
    - *Font:* It defines the color, face and size of a PC Data value.
  - ***Special entity with recursion:*** The following elements are special entities with recursion back to the inline entity:
    - *Anchor:* It is an anchor which links to some content. For example this entity is used to represent hyperlinks.
    - *Quote:* It is used to make a citation.

In addition the PSM has an identification entity which holds an id and the 'PIM'-id. It is used for the ground truth. As illustrated in figure 3.8 all the entities of the PSM inherit from the identification entity.
Figure 3.8 holds further entites which aren't explained in this section. Due to the fact, that the strucure of the PSM is quite similar to the HTML4 Specification deatails about them can be found in the specification.

3.3.6 Ground Truth model

The PIM and the PSM will be annotated with ground truths. They give a criterion to evaluate tools (e.g. characterization tools) against. Figure 3.10 illustrates the metamodel of the ground truth.

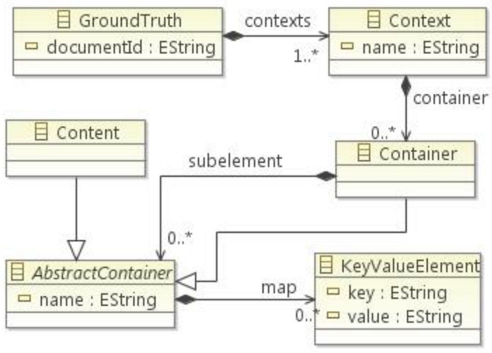


Figure 3.10: Ground Truth metamodel

Figure 3.10 shows that every ground truth has an Id which can be used to associate it with the corresponding PIM or PSM and holds a list of diverse contexts. A context has a name and contains a list of containers. These containers represent the ground truth for a specific context. For example the context is 'PIM'. Then the containers store information about the PIM.
The container entity is part of a composite pattern. This pattern is used because we need a possibility to represent the structure of a PIM or PSM with a tree. The ground truth meta-model illustrated in figure 3.10 contains an abstract container which holds a name and a list with key-value elements to describe features in detail. For example we know that a webpage contains three pictures then the key element is 'number of pictures of webpage' and the value is '3'. The content and container entity inherit from the abstract container. The container entity can hold a list of sub elements of type abstract container. This recursion is used to generate the structural trees of the PIMs and PSMs.

3.3.7 Relationships between the models

We introduced all the models which are part of our model driven benchmark data generation system. Moreover we gave a brief introduction into to the relationships between the models.
In the following section we want to point out the relationships of the models with focus on the model driven architecture. We introduced this software development methodology in chapter 2. The Model driven Architecture consists of a Computation Independent, Platform Independent and Platform Specific model. Through transformation these models can be mapped from CIM to PIM and to PSM.

Computation Independent Model (CIM)

The Computation Independent Model should define the *context, requirements, and purpose without any binding to computational implications [29]."* In our system the distribution model, genre model and vocabulary model can be seen as part of the CIM.

Property Distribution Model: It defines the required distributions of elements of the webpage which should be generated. Thereby it defines how often an image, video, hyperlink, paragraph and heading appear in a certain amount of webpages. The differentiation between paragraph and heading can be seen crucial, because it can be seen as part of the PSM because at the CIM level it can be seen as text. We do this distinction because we want to generate more practical webpages and we suppose that statistics also see them as different elements. Moreover the property distribution model can be seen as a requirement definition for the generated webpages.

Genre Model: The Genre model gives the user the possibility to define the type of webpages that should be generated. Thereby he can choose between a blog page and a simple page, but he doesn't decide about the concrete structure or the technical realization of it.
On the one hand the Genre model can be seen as part of the CIM because it defines only a proposal for the structure but on the other hand it is possible to argue that it defines the rough structure and is part of the PIM. We see the Genre Model as part of our CIM because it defines only requirements for our webpages. For example it defines the requirement that our webpages should look like a blog. Moreover this model is necessary to receive more practical webpages.

Vodabulary Model: The vocabulary model is some kind of data store. It holds pictures, videos, paragraphs, headings and hyperlinks. This vocabulary is hold as general as possible and should only support the creation of more practical webpages. Critics might see a problem because there is a differentiation between headings and paragraphs at the CIM level, but as mentioned above this distinction is necessary to receive the goal of generating more practical webpages. The vocabulary model can be seen as the definition of content.

Platform Independent Model (PIM)

As stated in chapter 2 the PIM should define the semantic structure without technology specific details. Therefore our PIM contains a content container, grid and listing entity for structuring our PIM. As real content entity the PIM provides a heading, paragraph, hyperlink, image and video.
By a look at the model the embedded Image and embedded Video seem really platform specific. The reason for modeling videos and images as embedded entities is the complexity of them. It would be really hard to model a video and an image as a PIM.
Every entity within the PIM holds only a small number of properties because the PIM should be hold platform independently as possible. For example the color can be defined or the size of entities.
If we have a look at our PIM we will find out, that it would be also possible to model other documents with it instead of webpages. This shows that the PIM is hold generally.

Platform Specific Model (PSM)

The Platform Specific Model is a model that defines all the platform and technology details. In our particular case we defined a PSM based on the HTML4 specification.
We followed a bottom-up development methodology by defining for all HTML4 functionalities an adequate feature in the PSM. For this reason the PSM covers the whole HTML4 functionalities except the features for dealing with forms. We ignored these features because they aren't necessary for our model driven generation framework. For this reason it is possible to model almost every HTML4 page with our PSM. Moreover it is possible to map features of the PIM to one or more features in the PSM.

Sample mappings

The following section provides three sample mappings between the models of the model driven architecture:

Example 1: We define at the CIM level that we want to generate one empty webpage. At the PIM level one PIM entity would be instantiated and at the PSM level a PSM entity with corresponding Head and Body Entity will be instantiated. If the PIM instance holds information about the author or date, meta entities must be instantiated to store this information. These entities are contained in the head entity. Figure 3.11 illustrates the mapping example.

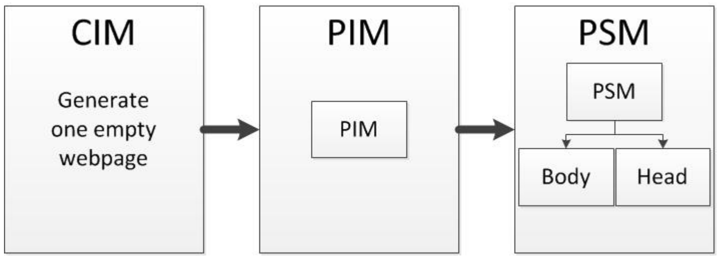


Figure 3.11: Example mapping empty webpage

Example 2: We define at the CIM level that we want to generate a webpage which contains one image. At the PIM level one PIM which holds an embedded image will be instantiated. And at the PSM level a PSM with corresponding head and body element would be instantiated. If the PIM instance holds information about the author or date, meta entities must be instantiated to store this information. These entities are contained in the head entity. The body element holds one flow entity and this element holds a graphic element. Figure 3.12 illustrates the mapping example.

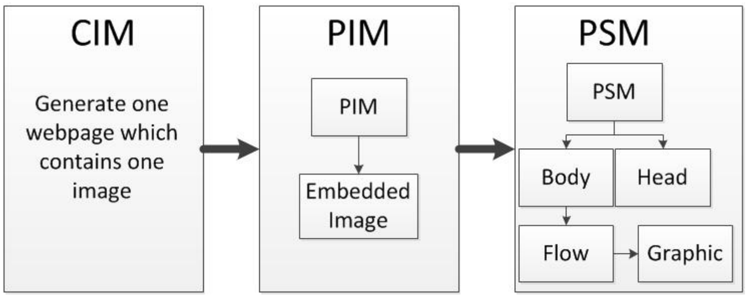


Figure 3.12: Example mapping image

Example 3: We define at the CIM level that we want to generate a webpage of type simple webpage. The page should be empty. At the PIM level a PIM will be instantiated which holds a grid with two grid rows and each row contains one grid cell. At the PSM level a PSM with corresponding head and body element would be instantiated. We have several possibilities to model the grid. We can model it as table, div containers or frames. If we model it with frames the body element instance must be removed. Figure 3.13 illustrates the mapping example. Moreover this examples outlines the power of the Model Driven Architecture. It is possible to have different mappings between a PIM entity and a PSM entity. Thus, it is possible to define different transformations and receive different results.

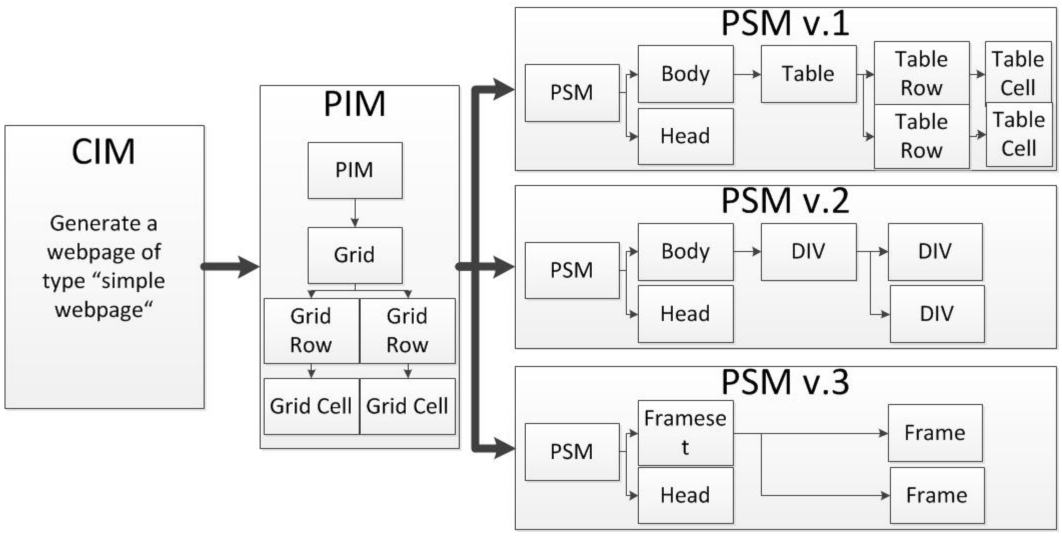


Figure 3.13: Example mapping webpage of type simple page

3.4 Technical Realization and Processes of the System

The following section provides insights into the technical realization of the models and transformations of the Model Driven Benchmark Data Generation Framework.

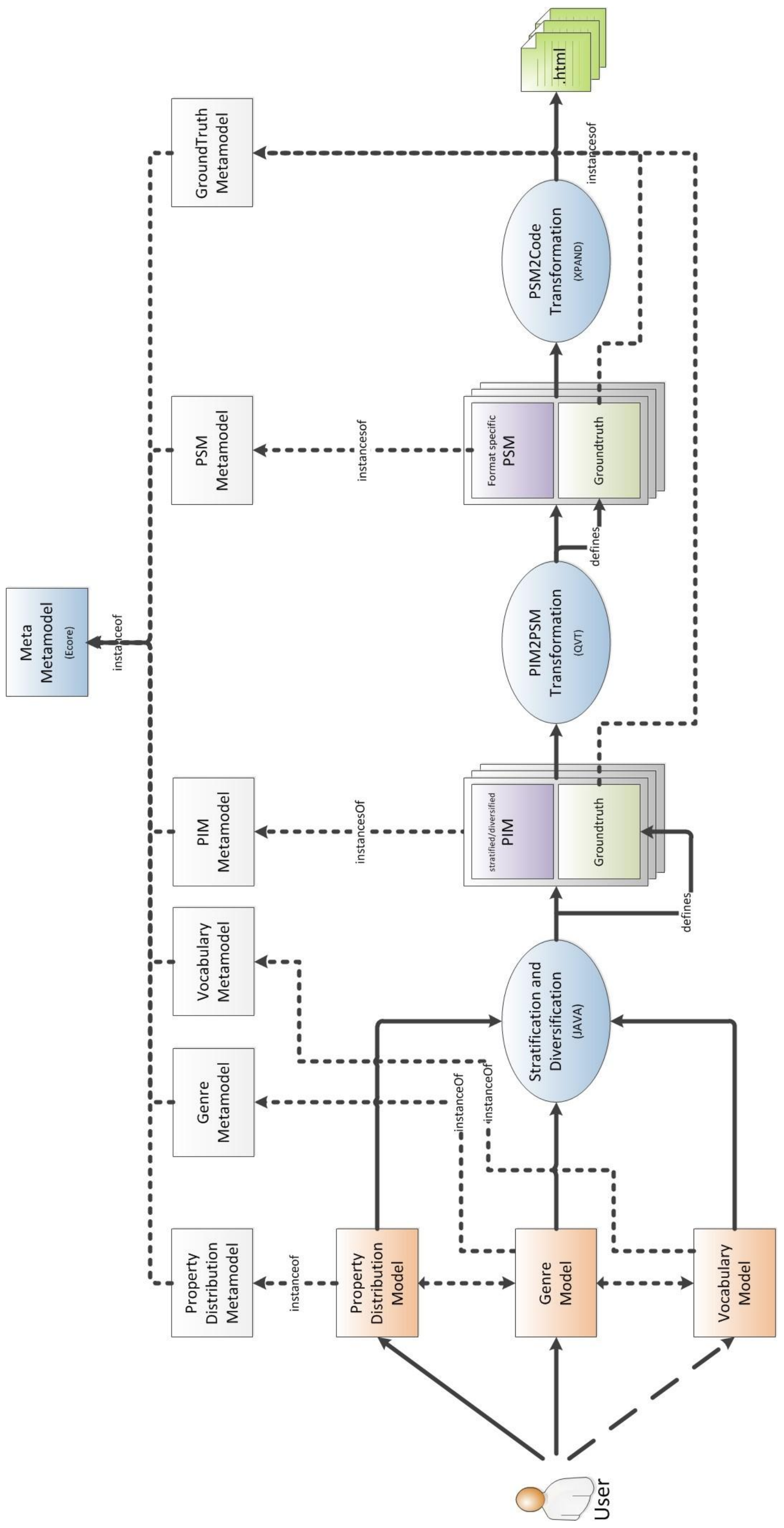


Figure 3.14: Model Driven Benchmark data generation framework workflow with technologies

3.4.1 Models

The system is based on the Eclipse Modeling Framework (EMF) and follows the methodology of Model Driven Architecture. As illustrated in figure 3.14 every model of our system has a meta-model. And every meta-model is an instance of the Ecore meta meta-model. Our system holds the following models with corresponding meta-models:

- Distribution model
- Genre model
- Vocabulary model
- Platform Independent model (PIM)
- Platform Specific model (PSM)
- Ground truth model

The Property Distribution, Genre and Vocabulary model can be instantiated or edited by the user. All the other models (PIM, PSM, Ground truth model) are instantiated and generated by the model driven benchmark data generation system.

3.4.2 Transformations

We introduced all the models of our system. These models are the basis of our system and they can be extended or edited. It is also possible to create new models and exchange or integrate them in our system. Besides, the models are only one part of the system because they can't be used without transformations. For example we have a Computational Independent Model and need a transformation to a Platform Independent Model and in a further step we need a transformation to a Platform Specific Model. Finally, it would be necessary to have a transformation from the Platform Specific Model to HTML code. This example illustrates the necessity of transformations. Therefore transformations are an important part of our system otherwise our approach would not work.
Before defining the transformations we should keep in mind the concrete use case for our benchmark data sets. As described in chapter 2 there are the following two use cases for a benchmark corpora:

- **System quality evaluation** (measure the correctness of tools and algorithms)
- **System performance evaluation** (should show the scalability of a tool or system)

As described in chapter 2 the benchmark corpora can be categorized by their purpose. For example for system quality evaluation a content-complete or feature complete corpus would be necessary. The combination of these two kinds of corpus can be a performance-defining corpus which can be used for performance evaluation.

Our approach is flexible because it is possible to generate benchmark corpora for quality evaluation or performance evaluation. For this reason it is necessary to define adequate transformations to generate the particular needed benchmark collection.
For example for system quality evaluation you can define a content-complete corpus. This corpus should cover *tthe widest range of content available in a given scenario [6].*" In our specific use case this would mean that all the content types covered by the statistics should be contained in our benchmark collection. Moreover this assumes that the statistics should be also content-complete.
In the following we explain the transformations and processes between our models and their technical realization.

Initial user definitions

Figure 3.14 shows that the first process of the Model Driven Benchmark Data Generation Framework consists of the initial user definitions. Thereby the user must instantiate a distribution model which is based on a frequency distribution and a genre model which defines the layout of the generated webpages. The user has also the possibilities to edit or enrich the predefined vocabulary model. All these models are available in XML format and based on the Eclipse Modeling Framework. For these definitions the Eclipse development environment can be used.

Diversification and Stratification process

The stratification and diversification process generates PIMs corresponding to the defined property distribution and genre model. It uses the vocabulary model to enrich the PIM features with contents.
The process can be divided into the following four sub processes:

1. The process reads the number of pages that should be generated from the property distribution model and creates as many PIMs as defined.

2. The created PIMs are filled with features based on the property distribution. All the inserted features are stored in a content container. For the contents of the features the process queries the vocabulary model. It receives contents randomly and inserts them to the features. Thereby every inserted feature gets a unique id.

3. The process structures the features of the PIMs based on the genre model.

4. Finally a ground truth for every PIM is generated. The ground truth holds a statistic about the number of headings, paragraphs, hyperlinks pictures, videos, listings and tables. It also contains a tree which represents the structure of the PIM. The tree stores every feature with corresponding id.

The whole process is implemented in Java and uses the EMF Java API. It is necessary to generate the model code of the distribution, genre, vocabulary, platform independent and ground truth model. This can be done with the Eclipse Modeling Framework.

PIM to PSM Transformation

The Model Driven Architecture transforms every PIM to a PSM. As illustrated in figure 3.14 our framework generate PSMs out of PIMs. We use the Query/View/Transformation (QVT) language to realize the model to model transformation. Therefore we had to define several mappings in QVT. Every element in the PIM had to be mapped to a corresponding element in the PSM.
Moreover this process generates a ground truth for every PSM. The ground truth holds statistics about the number of used HTML4-tags. It also contains a tree which represents the structure of the PSM. The tree holds PSM elements with unique ids and shows from which element in the PIM they are derived. This feature should enable the traceability of the ground truth.
For example figure 3.15 illustrates a PIM and a PSM structure tree and the relationship between the trees. The example assumes that there is a PIM instance which holds an embedded image. As described in chapter 3.3.4 every entity of a PIM holds a unique Id. The structure tree defines that our PIM instance holds a PIM entity with the Id 1 which contains an embedded image entity with Id 2. Due to the fact that the structure tree is a formal description of a PIM or PSM it does't contain concrete information about the entities of the PIM. For this reason the Ids can be used to find the concrete entities in the PIM instance. The blue arrows in figure 3.15 represent a transformation from PIM to PSM. Thereby the arrows show how the features are mapped. As mentioned above the ground truth for the PSM also holds a structure tree. In our concrete example the tree defines that our PSM holds a body and a head entity, and the body entity holds a graphic. The PSM description of the tree holds the concrete Id of the PSM entity in the PSM instance. Moreover it contains the Id of the entity in the PIM instance which has been transformed to the PSM entity. The same applies for the graphic entity description of the tree. The Body and Head entity description hold only the Ids of the concrete entities in the PSM instance. They don't contain a from id because they are created during the transformation.

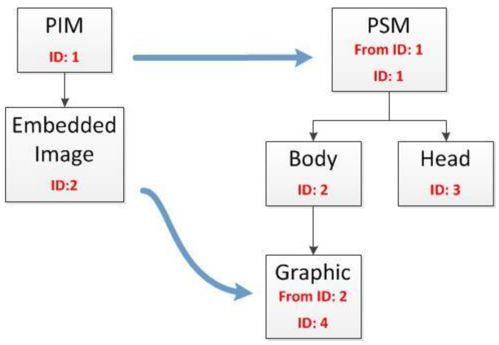


Figure 3.15: Structure tree example

PSM to Code Transformation

The last process of our framework is the transformation from the PSMs to Source Code. For every PSM an adequate HTML4 file will be generated. These files together with the ground truth for the PIM and the PSM are the main results of our model driven benchmark data generation

framework.
For the technical realization of the code generation we use Xpand which is a template based transformation language. For this reason we defined a template for HTML4 Code. This template will be used by Xpand to generate the code.

3.4.3 Automatic Workflow

One goal of the model driven benchmark data generation framework was that it works as automated as possible. At the beginning the user must define the distribution, genre and vocabulary model and then starts the system which delivers HTML source code as result. All the processes and transformations are done automatically.
We find a solution which integrates all the used technologies (Java, QVT, XPand) within one workflow, with each execution of the framework the number of generated pages can be changed. Therefore we had to find a way to deal with a dynamic number of pages to create.
Our Workflow can be divided into the following three workflow steps:

1. The diversification and stratification Ant-script is called. It generates an executable JAR file out of the java code for the diversification and stratification. Subsequent this file will be executed. Thereby it generates the PIMs with corresponding ground truth and the Ant-script for the PIM to PSM transformation. The generated Ant-script contains for every PIM a command for the transformation to a PSM. This dynamic generation of the Ant-script is necessary because the number of generated PIMs can vary.

2. The generated PIM to PSM transformation Ant-script is called and executed. It transforms the PIMs to corresponding PSMs.

3. The code generation ant script is called. At first it generates an Ant-script which contains for every PSM a command for the code generation. Afterwards this script will be executed and the source code generated.

The whole workflow can be executed with a Batch -File which is part of the framework. It calls the three workflow steps successive.

3.5 Summary

This chapter introduced the Model Driven Benchmark Data Generation Framework. The system is based on the Model Driven Architecture and uses the Eclipse Modeling Framework as underlying platform.
The system holds the following models:

- ***Distribution model:*** It is used for representing the property distributions of webpapes.

- ***Genre model:*** It defines the structure of a webpage.

- ***Vocabulary model:*** It is a kind of data store which holds the contents for the webpages.

- ***Platform Independent model:*** It is part of the model driven architecture and is a platform independent representation of a webpage.
- ***Platform Specific model:*** It is part of the model driven architecture and is a platform specific representation of a webpage.
- ***Ground Truth model:*** It represent the ground truth of a PIM or PSM.

And it can be divided in to the following four processes:

- ***Initial user definitions:*** The user must define a distribution, genre and vocabulary model.
- ***Stratification and diversification process:*** It generates PIMs with corresponding ground truths based on the models of the initial user definitions.
- ***PIM to PSM transformation:*** It transforms the PIMs to corresponding PSMs. Therby for every PSM a ground truth is generated.
- ***PSM to Code transformation*:** It transforms the PSMs to HTML4-Code.

For the PIM to PSM transformation it uses the QVT language and for the Code genreation Xpand.
The workflow is automated as possible and is created dynamical during each execution. It is based on ant scripts.
As result the systems delivers HTML4 webpages with corresponding ground truths for the PIMs and PSMs. These annotated benchmark data collections can be used for the evaluation of digital preservation tools.

CHAPTER 4

Evaluation

In this chapter we test the framework's outputs, do a clooser look on the specification of the transformations and measure the execution time. Finally, we evaluate the whole system based on the tests and criteria defined in the introduction of chapter 3.

4.1 Introduction

In the following chapter we are doing tests of the Model Driven Benchmark Data Generation Framework. For them we use a personal computer with the following characteristics:

- **Processor:** Intel Core Quad CPU Q8300 2,50GHz
- **Random Access Memory (RAM):** 4.00 GB
- **Operating System (OS)**: Windows 7 Professional 32Bit Copyright 2009 Microsoft Corporation

For the tests the following preperations are necessary:

- Downloading and Installing **Eclipse Modeling Tools (JUNO)** [1]
  - Installing the following Eclipse plugins:
    - ***Operational QVT***
    - ***XPAND***
- Installing **Java Runtime Environment 7** [2]

Before running the system and tests the following models must be instantiated:

[1] http://www.eclipse.org/downloads/packages/eclipse-modeling-tools/junor

[2] http://www.java.com/de/download/

- Property distribution model
- Genre model
- Vocabulary model

After these preperations we are able to run the system and our tests. We start the system with a batch file which is provided by the Model Driven Benchmark Data Generation Framework.

4.2 Tests of the system's outputs

The following test should evaluate the outputs of our system. The results of our system are generated webpages with corresponding ground truths for the PIM and PSM. We are doing the ouput tests manually. For this reason we can do them only for a small set of generated webpages. Our manually testing methodology is the following:

1. We define a property distribution model, genre model and vocabulary model.
2. We run the system to generate the HTML files with corresponding ground truths.
3. We evaluate the results:
   a) Evaluation of the generated webpages
   b) Evaluation of the PIM's ground truth
   c) Evaluation of the PSM's ground truth

4.2.1 Preparations for the test

For the evaluation of the results we generate ten web pages with the distributions illustrated by the following diagrams. They show the distribution of five features. For example figure 4.1 shows that 40 percent of ten pages have two headings and 60 percent of ten pages have one heading. All these distribution values refer to an amount of ten pages.

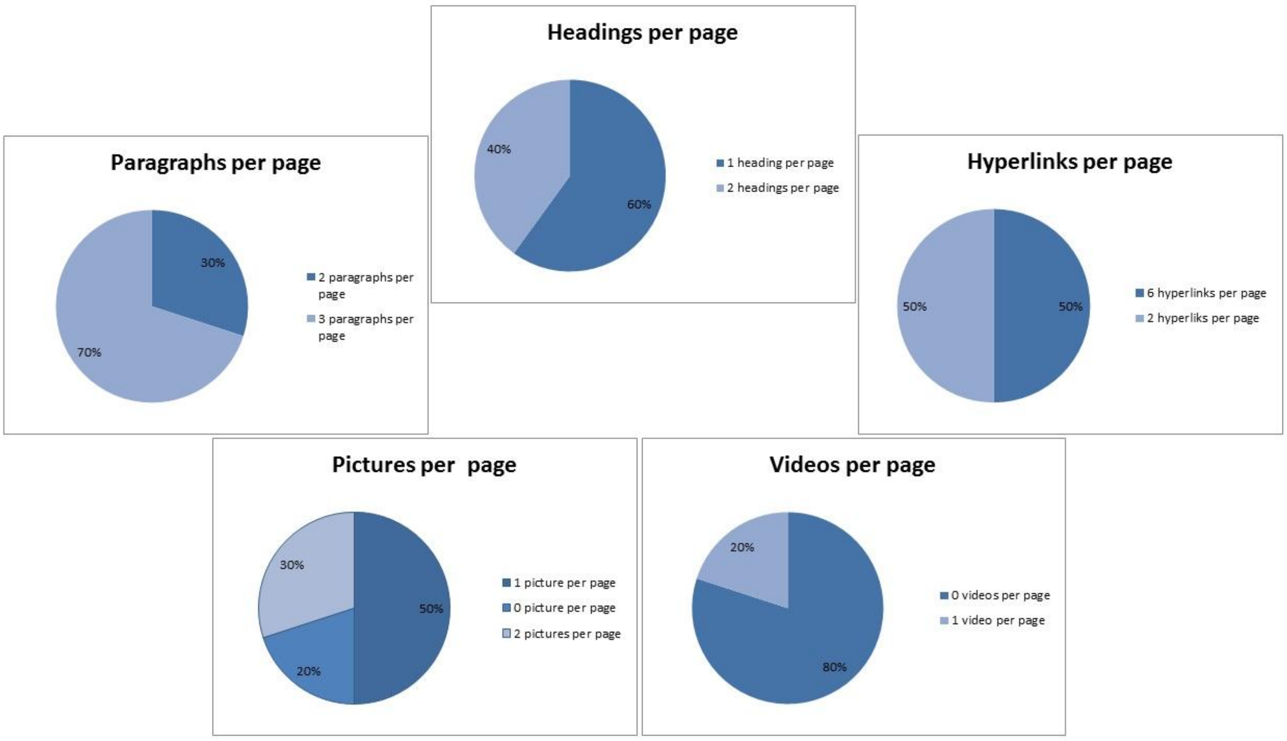


Figure 4.1: Property Distributions for the Evaluation

We define a genre model of type 'simple page' and with the content pattern illustrated by figure 4.2. This pattern defines the composition of elements in the content area. For our use case we take the following composition: Heading, Paragraph, Embedded Video, Paragraph and Embedded Image. This pattern can be repeated maximal six times.

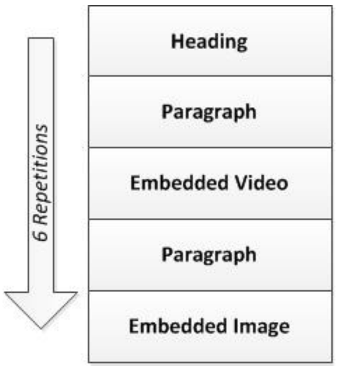


Figure 4.2: Content pattern

We use the predefined vocabulary model of the Model Driven Benchmark Data Generation

Framework. After defining the distribution, genre and vocabulary model we can run the system.

4.2.2 The test results

At first we evaluate the generated webpages for this reason we count the number of headings, paragraphs, pictures, videos, tables and listing of every page. By usage of this census we count the number of pages which contain the same amount of features. For example we count the number of web pages which hold one heading. In our case these are six web pages.
Due to the fact that we know the property distribution for the whole collection (10 webpages) we can calculate the expected number of webpages holding a certain characteristic. Then we can compare this value with the counted number. If the expected and counted value is the same we can obtain that the system worked correctly. For example 60 percent of webpages have one heading which means that the expected number of pages is six. As we observed above the counted number of webpages holding one heading is six. If we compare these two values we come to the conclusion that they are the same and the system worked correctly.
We apply this comparison methodology to all the features of our distribution and come to the conclusion that the system generates all of them as defined. The results are covered in table 4.1.

	Expected percentage of pages	**Expected number of pages**	**Counted number of pages**
Headings			
1 heading per page	60%	6	6
2 headings per page	40%	4	4
Paragraphs			
2 paragraphs per page	30%	3	3
3 paragraphs per page	70%	7	7
Hyperlinks			
2 hyperlinks per page	50%	5	5
6 hyperlinks per page	50%	5	5
Pictures			
0 picture per page	20%	2	2
1 picture per page	50%	5	5
2 pictures per page	30%	3	3
Videos			
0 video per page	80%	8	8
1 video per page	20%	2	2

Table 4.1: Evaluation of the generated webpages

In a further step we evaluate the Ground truth of the PIMs. This ground truth holds a statistic about the number of features of every PIM. Therefore we count the number of features of every PIM and compare it with the ground truth value. We figure out that the counted and the ground truth value are always the same. Therefore we obtain that the PIM's ground truth covers the right statistics. Finally, we evaluate the ground truth of the PSMs. It holds a statistic about the used HTML-tags. For this reason we count every HTML Tag of every web page and compare these values with the values of the ground truth. We find out that also the PSM's ground truth covers the HTML tags correctly.

4.3 Transformations

As described in chapter 3 the core elements of the Model Driven Benchmark Data Generation Framework are transformations. There are several transformations which are used in our framework. Due to the fact that we use the model driven architecture we have the following transformations:

- **Computational Independent Model (CIM) to Platform Independent Model (PIM)**
- **Platform Independent Model to Platform Specific Model (PSM)**
- **Platform Specific Model to Code**

In the following section we explain these transformations and their capabilities.

4.3.1 Computational Independent Model (CIM) to Platform Independent Model (PIM)

In chapter 3 we showed that this transformation is done by the Diversification and Stratification Process. This process uses the Genre, Vocabulary and Distribution model as input and generates corresponding PIMs. The Distribution model can be seen as a requirement definition. As described in chapter 3 it defines the distribution of features of a certain amount of webpages. Moreover the Genre and Vocabulary model stay in dependency to the distribution model. For example the Vocabulary model should deliver contents for all the features defined by the Distribution model. The genre model specifies the kind of webpage and the sequencing of the features defined by the Distribution model.
Table 4.2 shows the transformation from requirements (CIM) to the PIM. The left column of the table shows entities on CIM level and the right column shows the corresponding entities of the PIM. The transformation is done by the diversification and stratification process which is a Java program.
At the Diversification and Stratification process for every webpage a PIM entity with a corresponding size entity is instantiated. The transformation generates the semantic structure of the page. Therefore it uses the specified page types of the genre model. For example the process generates for the page type simple page or blog, one grid which contains two more grids. These grids hold the contents of the PIM.

The contents can be headings, videos, hyperlinks, images or videos. The mapping of these entities is described by table 4.2. For example an image is transformed to an embedded image.

Entity at CIM level	Entity at PIM level
-	PIM + Size
Heading	Heading + Color
Paragraph	Paragraph + Color
Hyperlink	Hyperlink + Color
Image	Embedded Image + Size
Video	Embedded Video + Size
-	Grid
-	Grid Row
-	Grid Cell
-	Border
-	Listing
-	Listing Point
-	Content Container
-	Identification

Table 4.2: CIM to PIM Transformation

Sometimes the table contains at the left column any entry but on the right column a corresponding. This means that the feature doesn't exist on CIM level but on PIM level.

4.3.2 Platform Independent Model to Platform Specific Model (PSM)

The Model Driven Architecture specifies that every PIM must be transformed in an adequate PSM. This transformation is done by QVT. Table 4.3 gives an overview of one possible transformation from PIM to PSM. Thereby the left column shows the PIM entities and the right column the corresponding PSM entities. Every entity from the right column is transformed to one or more corresponding entities on the left column. Thereby all the attributes from the PIM entity are transformed to attributes of the corresponding PSM entites.
For example a concrete PIM entity instance holds a title, documentId, author, date attribute and an array of keywords. Table 4.3 shows that the PIM entity is mapped to a PSM entity which holds a Head and Body entity. In the Head element the metadata (e.g.keywords, documentId, author, date) and the title are stored. The Body entity holds the mapped contents.
For example as illustrated in table 4.3 a Paragraph entity at the PIM level is mapped to a Paragraph which holds a PC Data entity at the PSM level. This means that the Paragraph entity at the PSM level specifies the kind of element and the value attribute (=content) of the PIM's paragraph is stored in the PC Data entity.
Our transformation doesn't map to all entities of the PSM because we don't need them. It is possible that an entity of the PIM can be transformed into diverse entities of the PSM. We outlined

this topic in chapter 3 where we pointed out the relationships between the models. We showed that a grid can be transformed into a table, DIV container or a frameset. This fact shows that it is possible to define different transformations. In our particular case we defined that a grid is transformed into a table. For example we can define another transformation where a grid is transformed into a div container or a frameset.

Entity at PIM level	**Entity at PSM level**
Identification	Identification
PIM	PSM, Head (Meta), Body
Color	Color
Content Container	Body Element Collection
Grid + Border	Table
Grid Row	Table Row
Grid Cell	Table Cell
Listing	Ordered List/Bullet List
Listing Point	List Item
Paragraph	Paragraph + Pc Data
Heading	Heading + Pc Data
Hyperlink	Anchor + Pc Data
Embedded Image	Graphic
Embedded Video	Object + Param Collection (Embed, Param)

Table 4.3: PIM to PSM Transformation

4.3.3 Platform Specific Model to Code

Finally, the PSM is transformed to code. As mentioned in chapter 3 we based our PSM on the HTML4 specification. We provide a XPand template which transforms every entity of the PSM to Code.

4.4 Tests of the execution Time

For the evaluation of the execution time of our Model Driven Benchmark Data Generation Framework we use the distribution of the output tests and vary the number of pages. We also use the genre and vocabulary model from the previous test.
Our testing methodology is the following:

1. We edit the number of pages in the distribution model.

2. We use the Windows PowerShell and the Measure-Command [3]. This command measures the running time of a script or command. With these command we measure the execution time of the:

    - Diversification and stratification process
    - PIM to PSM transformation process
    - PSM to Code transformation process

3. After the execution of each process the measure command delivers us the runtimes. We take a note of them.

We repeat our methodology until we have done it for the following numbers of pages: 10, 25, 50, 75, 100, 250, 500, 750, 1000, 2500 and 5000.
Table 4.4 shows the results of the tests in milliseconds. We obtain the overall time through adding the runtime of the diversification and stratification, the PIM to PSM transformation and PSM to Code transformation process.

Number of Pages	**Diversification and stratification**	**PIM to PSM transformation**	**PSM to Code transformation**	**Overall time**
10	4879,4	19892,4	5140,7	29912,5(29,9 s)
25	5208,1	33099,5	5594,5	43902,1 (43,9 s)
50	5299,8	54509,1	6165,9	65974,8 (1,1 min)
75	5567,1	75882,8	6938,1	88388,0 (1,47 min)
100	5815,5	97346,5	7364,8	110526,8 (1,84 min)
250	7150,1	229466,1	11026,1	247642,2 (4,13 min)
500	9510,0	447002,1	18798,2	475310,3 (7,92 min)
750	11757,3	678738,9	26647,4	717143,6 (11,95 min)
1000	14865,2	9064256,0	34960,3	956251,5 (15, 94 min
2500	28844,5	2446167,5	87177,7	2562189,7 (42,7 min)
5000	60961,6	6575534,6	220934,8	6857431,0 (114,29 min)

Table 4.4: Resulsts of the execution time measurement (in milliseconds)

Figure 4.3 visualizes the measured values of table 4.4. The X-axis shows the number of pages and the Y-axis holds the runtime in milliseconds.

[3]http://technet.microsoft.com/en-us/library/ee176899.aspx

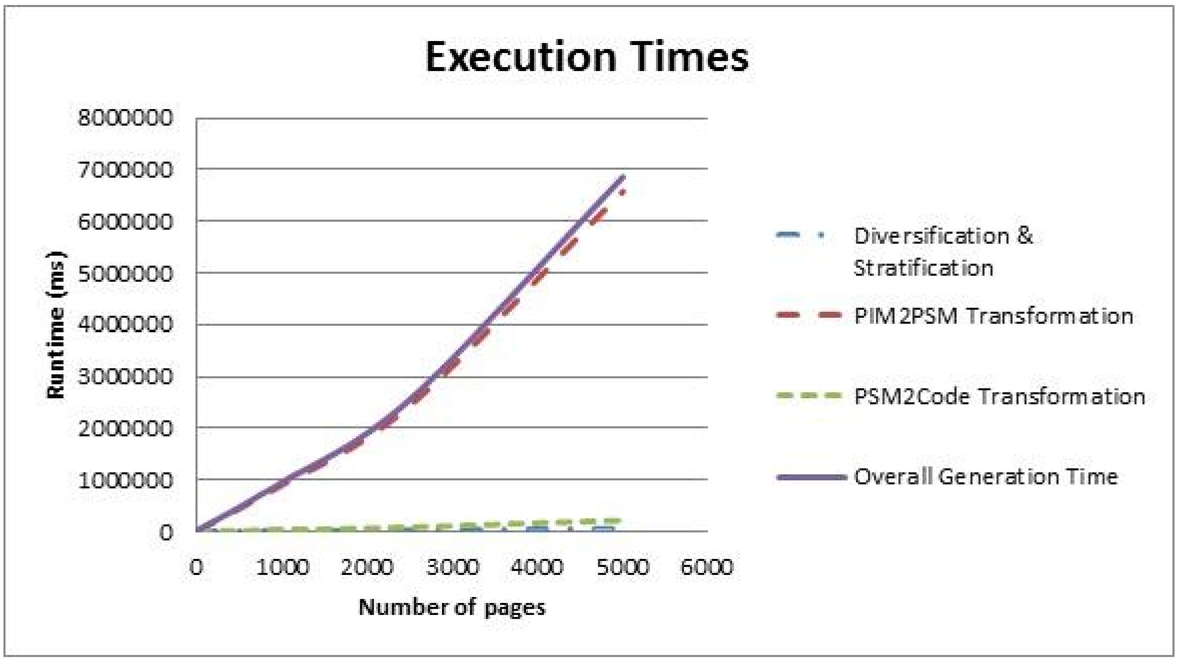


Figure 4.3: Diagram of the exectuion time

The dashed and dotted line (blue) represents the duration of the diversification and stratification process. This process is based on a Java program which uses the EMF API. In all test cases this process has the lowest runtime. It is increasing slowly.
The dashed (big dashes) line (red) illustrates the runtime of the PIM to PSM transformation. This transformation is based on QVT. It transforms the PIMs to PSMs and generates corresponding ground truths for the PSMs. It is the process with the longest duration and increases nearly linear. The reason for this is QVT and the fact, that this transformation makes a one-to-many mapping which is complex. This means that it takes one model as input and has two models as output.
The dashed (small dashes) line (green) represents the PSM to Code transformation process. It uses XPAND as Technology and increases slowly.
Finally it is to mention that the continious line (purple) represents the overall generation time of the system. It is interesting that it increases nearly linear and quite similar as the PIM to PSM transformation.

4.5 Evaluation of the system

Our developed model driven benchmark data generation framework shows a way to solve the problem of the inexistence of annotated benchmark data in digital preservation. As shown in the tests we have the ability to generate webpages with corresponding ground truth. Thereby we follow a new approach through generating them based on property distributions.

We tested the outputs of our system to make a decision about the correct functionality of the system. We generated ten webpages which were based on a sample property distribution. The tests showed that the system works as expected. The system generates the right amount of webpages based on a certain property distribution. Moreover we evaluated the ground truths and came to the conclusion that they are correct.
In a further step we evaluated the execution times of our system and how the particular transformation processes perform. We generated from ten up to 5000 pages. Moreover these tests should show the scalability of the system. After a detailed analysis of the different execution times of the processes we found out that the overall execution time increases nearly linear and is almost the same as the PIM to PSM transformation's execution time. This fact shows that the execution time of the Stratification and Diversification and the code generation process is not significant for the overall execution time. The PIM to PSM transformation is a really time-consuming process and affects the overall execution time. Moreover these tests showed that the system scales and can handle a huge workload.
We analyzed the transformations between the different abstraction levels of the model driven architecture in detail. At first we had a look on the diversification and stratification process which can be seen as the Computation Independent Model (CIM) to Platform Independent Model (PIM) transformation. Based on a certain property distribution the PIMs are instantiated. These PIMs are enriched by contents of the vocabulary model and structured based on requirements defined by the genre model. During this transformation a lot of entities of the PIMs are instantiated or generated because they don't appear at the CIM level.

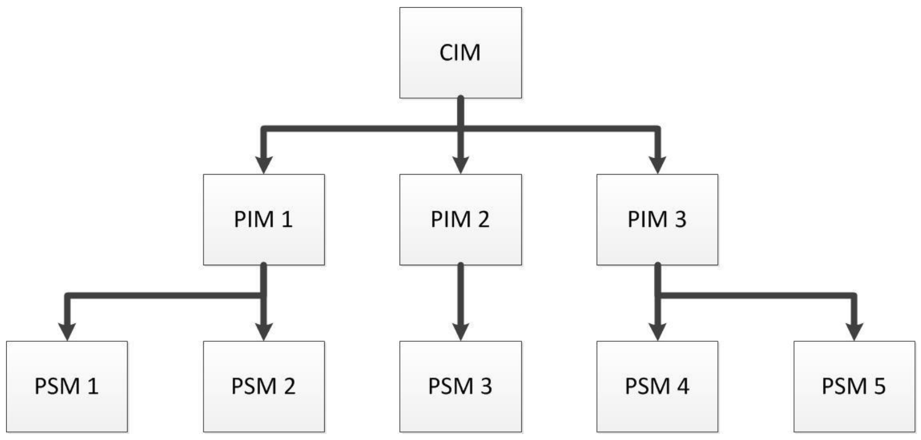


Figure 4.4: Power of the MDA

The PIMs are transformed to Platform Specific Models (PSMs). Therefore a transformation must be specified. This transformation contains several transformation rules which map one PIM entity to an adequate entity of the PSM. A PIM entity can mapped to different PSM entities. For this reason it is possible to define different transformations between the PIMs and PSMs. For example a grid can be mapped to a table, div container or frameset.
This example demonstrates the power of the Model Driven Architecture methodology. In fig-

ure 4.4 we explain the power of this methodology. We assume that we have one CIM which is mapped to three different PIMs and the PIMs are transformed to different PSMs. For example out of one CIM different types of webpages can be generated. It would be possible to generate HTML3 and HTML4 based webpages.
The following question appears: What is the human effort to specify new transformations? The definition of them implies that the developer is familiar with using EMF transformation languages. Our system uses Java, QVT and Xpand. If the developer wants to define a new transformation it is necessary that he defines a transformation rule for every entity in the source model. The developer has two possibilities he can define a new transformation file or edit an existing one. The effort depends on the number of transformation rules he wants to edit or create. It is important that every entity in the source model is mapped to an entity in the target model. This is necessary because otherwise the coverage of all entities of a source model would not be given. It isn't important that the transformation rule maps to all the entities of a target model because it can hold a lot more entities than the source model. At this point the question appears about the general coverage of the models of the system. The coverage depends on the relevance of certain features for a benchmark corpus. Our prototype holds the core properties of a webpage (Paragraphs, Headings, Hyperlinks, Tables, Listings, Pictures, Videos) but he did not cover all the properties of a webpage. The PIM covers all the mentioned core features and the PSM covers the whole HTML4 Specification without form entities. At the PIM level all the features are used but at the PSM level only a few are used. The necessity of a higher rate of coverage depends on the relevance of the attributes for a benchmark corpus.
As we evaluated so far the system fulfills our expectations but we will do a closer look on it with respect to the criteria we defined in chapter 3.
Our primary goal was to find a way to generate webpages with corresponding ground truth. With our framework we are in the position to generate HTML 4 based webpages. This fact shows the first limitation of our system, because there are a lot of diverse formats for representing webpages and we support only one. We accepted this limitation because we focused more on the stratification and diversification process. Due to the fact that we use the model driven architecture it is possible to extend our system for generating other kinds of webpages.
Our generated webpages hold two ground truths. As stated in chapter 3 the ground truths should support traceability, describe the structure and properties of a webpage. Our ground truths are able to solve all of these criteria. We can represent the structure of our webpages with a tree and store the ids of the properties. Therefore we can retrace from where a certain feature comes from.
As mentioned above the main focus was on the Stratification and Diversification process. As stated in chapter 3 one of our goals was the generation of more realistic webpages. As basic for this we defined the Genre, Vocabulary and Property distribution model. Thereby the genre model defines the layout, the vocabulary model is a data store for the contents of our webpages and the Property Distribution model holds the statistically values of webpage features. As mentioned in chapter 3 there are dependencies between these models which are hard to prevent because the three models stand in connection to each other. We accepted these dependencies because it provided us an approach to generate the webpages based on property distributions which is one of the criteria defined in chapter 3. Without the Vocabulary model we had to face the problem

from where we get the contents of our webpages and without the genre model it had been really hard to define the structure of a webpage.
One criteria of our benchmark data generation framework was to solve the challenges of benchmark corpora defined in chapter 2. In the following we describe how these challenges can be solved:

- **Precise Task Definition:** As described in chapter 3 our system can be used for system quality evaluation and system performance evaluation. Therefore the concrete task depends on the property distribution which defines the amount of pages and their properties. Moreover the coverage of the corpus depends on the amount of features covered by the statistically distributions.
- **Size of Corpus:** The size of the corpus depends on the precise task and the property distribution. The model driven benchmark data generation framework can generate a small or large set of webpages with corresponding ground truth.
- **Stratification:** The stratification is guaranteed because of the usage of property distributions for the generation of the benchmark corpora. Thereby the coverage of features depends on the distributions hold by the distribution model
- **Data Representation:** The generated webpages are represented in HTML4 which is a machine readable and process able form. Moreover the ground truths are represented by XML files which can be interpreted by a machine or human.
- **Ground Truth:** As mentioned above every generated webpage holds two ground truths where the results can be evaluated against.

Our system is based on the Model Driven Architecture and uses the EMF. Due to this fact we satisfy several criteria. We don't rely on proprietary formats and use a well-known technology for our system. We also provide a system which is easy to install and use if the user has a little bit knowledge about the EMF. Moreover we facilitate the execution of the system with a batch file. It calls the automatic workflow which was also one of the criteria stated in chapter 3.

4.6 Summary

At first we did the evaluation of the results of the Model Driven Benchmark Data Generation Framework and found out, that it works as espected. We manualy evaluated the outputs of generating ten webpages.
We showed the specification of the transformations of the framework.
We measured the execution time of each process and obtained the overall execution time. We came to the conclusion that the execution time depends strongly on the PIM to PSM transformation.
Finally, we evaluated the whole system based on our tests and the criteria defined in chapter 3. We showed that the system in general fulfills all the criteria with a few limitations.

CHAPTER 5

Conclusion and Outlook

This thesis introduced a solution to solve the problem of missing benchmark data with corresponding ground truth in Digital Preservation. These benchmark data collections are necessary to test and evaluate approaches of Digital Preservation.
Research in the field of Digital Preservation outlines the necessity of benchmark data collections. For example the National Library of Australia outlines in [7] that there is a big problem with the evaluation of preservations tools because of the inexistence of annotated benchmark data collections. In [7] they came to the conclusion that without knowing the correct answer of a certain test data, the correctness of a tool is not possible to determine. C. Becker et al state that *"annotated benchmark data are needed to support the objective comparison of new approaches and quantify the improvements over existing technologies [5]."* [6] and [7] outline that it is necessary to have metrics for evaluating the correctness of tools.
The goal of this work was to find, implement and evaluate a new approach to generate benchmark data with corresponding ground truth. The solution had to work as automated as possible and guarantee the scalability.
As development methodology we used the model driven architecture which enables the usage of the Eclipse Modeling Framework (EMF) which is a well-known software engineering technology. We extended the EMF with QVT for the model to model transformations and XPand for the code generation.
We developed a model driven benchmark data generation framework which follows a new approach. Instead of analyzing existing data collections, like it is done in Digital Forensics, Information Retrieval and Machine Learning, we generate our benchmark data from scratch.
Shannon stated in [46] that there exists a statistical distribution how often a letter appears within a message. We mapped Shannon's theory of entropy of letters to our challenge of generating webpages. Instead of message we have webpages and instead of letters we have features of webpages. Thus, we assume that every webpage has a distribution how often a feature appears. We use these distributions to generate our webpages.
Our benchmark data generation framework executes the following four processes sequentially:

1. **Initial User Definitions:** The user must define the property distribution in a distribution model, the layout of the webpages with a genre model and the contents to insert with a vocabulary model.

2. **Stratification and Diversification:** This process generates the Platform Independent Models (PIM) based on the property distribution, genre and vocabulary model. Thereby for every PIM a ground truth with a description of the structure and a statistic about the features is generated.

3. **PIM to PSM Transformation:** This process transforms the PIMs to Platform Specific Models (PSM). Thereby a ground truth with a description of the structure and a statistic about the used HTML Tags is generated.

4. **PSM to Code Transformation:** This process generates HTML4 Code based on the PSMs.

As the evaluation of the system has shown the system delivers the expected results. It delivers HTML4 pages based on property distributions and ground truths for the PIMs and PSMs. These ground truths hold statistically values and structure trees.
One limitation of the system is the generation of only HTML4 based webpages. Due to the fact that we use the Model Driven Architecture it would be possible to extend the system to further webpage formats. This limitation has been accepted because the focus of the system was on the Stratification and Diversification process.
The evaluation showed that the system scales and that the execution time is acceptable. As mentioned above the system is extendable and is easy to execute because it provides a Batch file which activates the system's automatic workflow.
In the future this system will be used to generate large benchmark data collections. It would be possible to extend the system that it generates a large variety of HTML formats. These benchmark data collections will be used to test and benchmark approaches, methods, technologies, frameworks in the field of digital preservation.

Bibliography

[1] Colin Webb. *Guidelines for the preservation of digital heritage*. Information Society Division, United Nations Educational, Scientific and Cultural Organization, 2003.

[2] David Reinsel, Christopher Chute, Wolfgang Schlichting, John McArthur, Stephen Minton, Irida Xheneti, Anna Toncheva, and Alex Manfrediz. The expanding digital universe. *White paper, IDC*, 2007.

[3] Digital Preservation Coalition. *Preservation Management of Digital Materials: The handbook*. Digital Preservation Coalition and National Library of Australia, York (UK), 2008.

[4] Helen Heslop, Simon Davis, and Andrew Wilson. *An approach to the preservation of digital records*. National Archives of Australia Canberra, 2002.

[5] Christoph Becker and Andreas Rauber. Decision criteria in digital preservation: What to measure and how. *Journal of the American Society for Information Science and Technology*, 62(6):1009–1028, 2011.

[6] Robert Neumayer, Hannes Kulovits, Andreas Rauber, Manfred Thaller, Eleonora Nicchiarelli, Michael Day, Hans Hofman, and Seamus Ross. On the need for benchmark corpora in digital preservation. In *Proceedings of the 2nd DELOS Conference on Digital Libraries*, 2007.

[7] Matthew Hutchins. *Testing Software Tools of Potential Interest for Digital Preservation Activities at the National Library of Australia*. National Library of Australia, 2012.

[8] Paul Conway. *Preservation in the digital world*. Commission on Preservation and Access Washington, DC, 1996.

[9] Terry Kuny. The digital dark ages? challenges in the preservation of electronic information. *International Preservation News*, pages 8–13, 1998.

[10] ISO 1472:2012. Space data and information transfer systems - open archival infromation system (oais)- reference model. 2012.

[11] Stephan Strodl, Petar Petrov, and Andreas Rauber. Research on digital preservation within projects co-funded by the european union in the ict programme. *Vienna University of Technology, Tech. Rep*, 2011.

[12] Donald Waters and John Garrett. *Preserving Digital Information. Report of the Task Force on Archiving of Digital Information.* ERIC, 1996.

[13] Ulla Bøgvad Kejser, Anders Bo Nielsen, and Alex Thirifays. Cost model for digital preservation: Cost of digital migration. *International Journal of Digital Curation*, 6(1):255–267, 2011.

[14] Christoph Becker, Hannes Kulovits, Michael Kraxner, Riccardo Gottardi, Andreas Rauber, and Randolph Welte. Adding quality-awareness to evaluate migration web-services and remote emulation for digital preservation. In *Research and Advanced Technology for Digital Libraries*, pages 39–50. Springer, 2009.

[15] Feng Luan, Thomas Mestl, and Mads Nygård. Quality requirements of migration metadata in long-term digital preservation systems. In *Metadata and Semantic Research*, pages 172–182. Springer, 2010.

[16] Sarika Sawant. Strategies for preservation of digital documents: An overview. *Trends in Information Management (TRIM)*, 2(1), 2012.

[17] Klaus Rechert, Dirk von Suchodoletz, and Randolph Welte. Emulation based services in digital preservation. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 365–368. ACM, 2010.

[18] Klaus Rechert, Dirk von Suchodoletz, and Isgandar Valizada. Migration-by-emulation planets web-service. 2011.

[19] Stephan Strodl, Andreas Rauber, Carl Rauch, Hans Hofman, Franca Debole, and Giuseppe Amato. The delos testbed for choosing a digital preservation strategy. In *Digital Libraries: Achievements, Challenges and Opportunities*, pages 323–332. Springer, 2006.

[20] Brian Aitken, Seamus Ross, Andrew Lindley, Edith Michaeler, Andrew Jackson, and Maurice Van Den Dobbelsteen. The planets testbed. In *Research and Advanced Technology for Digital Libraries*, pages 401–404. Springer, 2010.

[21] Brian Aitken, Petra Helwig, AN Jackson, Andrew Lindley, Eleonora Nicchiarelli, and Seamus Ross. The planets testbed: Science for digital preservation. *Code4Lib Journal*, 1(5), 2008.

[22] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[23] Ellen Voorhees, Donna K Harman, et al. *TREC: Experiment and evaluation in information retrieval*, volume 63. MIT press Cambridge, 2005.

[24] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, pages 3–10, 2007.

[25] Thomas Gloe and Rainer Böhme. The dresden image database for benchmarking digital image forensics. *Journal of Digital Forensic Practice*, 3(2-4):150–159, 2010.

[26] Akira Yanagawa, Alexander C Loui, Jiebo Luo, Shih-Fu Chang, Dan Ellis, Wei Jiang, Lyndon Kennedy, and Keansub Lee. Kodak consumer video benckmark data set: concept definition and annotation. *Columbia University ADVENT Technical Report*, pages 246–2008, 2008.

[27] Helge Homburg, Ingo Mierswa, Bülent Möller, Katharina Morik, and Michael Wurst. A benchmark dataset for audio classification and clustering. In *ISMIR*, volume 2005, pages 528–31, 2005.

[28] J Stephen Downie. The music information retrieval evaluation exchange (mirex). *D-Lib Magazine*, 12(12):1082–9873, 2006.

[29] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering*, 1(1):1–182, 2012.

[30] Joaquin Miller, Jishnu Mukerji, et al. Mda guide version 1.0. 1. *Object Management Group*, 234:51, 2003.

[31] Anneke G Kleppe, Jos Warmer, and Wim Bast. The model driven architecture: practice and promise, 2003.

[32] Mohsen Asadi, Mahdy Ravakhah, and Raman Ramsin. An mda-based system development lifecycle. In *Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference on*, pages 836–842. IEEE, 2008.

[33] Martin Kempa and Zoltán Adám Mann. Model driven architecture. *Informatik-Spektrum*, 28(4):298–302, 2005.

[34] Thomas O Meservy and Kurt D Fenstermacher. Transforming software development: An mda road map. *Computer*, 38(9):52–58, 2005.

[35] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.

[36] Frédéric Jouault and Ivan Kurtev. On the architectural alignment of atl and qvt. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 1188–1195. ACM, 2006.

[37] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, Ivan Kurtev, and Patrick Valduriez. Atl: a qvt-like transformation language. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 719–720. ACM, 2006.

[38] Qvt Omg. Meta object facility (mof) 2.0 query/view/transformation specification. 2011.

[39] Rhys Lewis. Glossary of terms for device independence. *W3C Note*, 2005.

[40] Zlatko Čović, Miodrag Ivković, and Biljana Radulović. Mobile detection algorithm in mobile device detection and content adaptation. *Acta Polytechnica Hungarica*, 9(2):95–113, 2012.

[41] Velibor Adzic, Hari Kalva, and Borko Furht. A survey of multimedia content adaptation for mobile devices. *Multimedia Tools and Applications*, 51(1):379–396, 2011.

[42] Timo Laakko and Tapio Hiltunen. Adapting web content to mobile user agents. *Internet Computing, IEEE*, 9(2):46–53, 2005.

[43] Jiang He, Tong Gao, Wei Hao, I-Ling Yen, and Farokh Bastani. A flexible content adaptation system using a rule-based approach. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):127–140, 2007.

[44] Juan F Lope and Pedro Szckeiy. Web page adaptation for universal access. 2001.

[45] YD Yang, JL Chen, and HJ Zhang. Adaptive delivery of html contents. *WWW9 Poster Proceedings*, pages 24–25, 2000.

[46] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.