Enhanced Presentation and Machine-Understandable Metadata for Digital Comics using Open Web Formats

Pieter Heyvaert

Supervisor: Prof. dr. ir. Rik Van de Walle

Counsellors: Ir. Miel Vander Sande, Ir. Tom De Nies, Dr. Wesley De Neve, Joachim Van Herwegen

Master's dissertation submitted in order to obtain the academic degree of Master of Science in de ingenieurswetenschappen: computerwetenschappen

Department of Electronics and Information Systems
Chairman: Prof. dr. ir. Jan Van Campenhout
Faculty of Engineering and Architecture
Academic year 2013-2014
Enhanced Presentation and Machine-Understandable Metadata for Digital Comics using Open Web Formats

Pieter Heyvaert

Supervisor: Prof. dr. ir. Rik Van de Walle

Counsellors: Ir. Miel Vander Sande, Ir. Tom De Nies, Dr. Wesley De Neve, Joachim Van Herwegen

Master’s dissertation submitted in order to obtain the academic degree of Master of Science in de ingenieurswetenschappen: computerwetenschappen

Department of Electronics and Information Systems
Chairman: Prof. dr. ir. Jan Van Campenhout
Faculty of Engineering and Architecture
Academic year 2013-2014
Preface

The goal of this thesis is to guide publishers and authors into a new era for digital comics. The work conducted solves problems caused by the current (proprietary) formats for digital comics and tries to unify their features in a standard for electronic publications, called EPUB 3.

I would like to thank my parents for supporting my educational voyage at Ghent University over these past five years, my friends for the distraction in between the studying, the projects and this thesis. A word of thanks is for my promotor Rik Van de Walle and my supervisors Wesley De Neve, Tom De Nies, Joachim Van Herwegen and Miel Vander Sande for all their help during my work conducted for this thesis. It would not be in the form it is in today if it were not for them aiding me when I stumbled upon all types of hiccups. Finally, I would like to thank WPG Uitgevers België and Ballon Media for providing me with representative content for conducting my work.

Pieter Heyvaert, May 2014
Permission for Usage

“The author gives permission to make this master dissertation available for consultation and to copy parts of this master dissertation for personal use. In the case of any other use, the limitations of the copyright have to be respected, in particular with regard to the obligation to state expressly the source when quoting results from this master dissertation.”

Toelating tot bruikleen

“De auteur geeft de toelating deze masterproef voor consultatie beschikbaar te stellen en delen van de masterproef te kopieren voor persoonlijk gebruik. Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze masterproef.”

Pieter Heyvaert, May 2014
Enhanced Presentation and Machine-Understandable Metadata for Digital Comics using Open Web Formats

by

Pieter Heyvaert

Master’s dissertation submitted in order to obtain the academic degree of Master of Science in de ingenieurswetenschappen: computerwetenschappen

Academic year 2013-2014

Ghent University
Faculty of Engineering and Architecture
Department of Electronics and Information Systems (ELIS)
Chairman: Prof. Dr. Ir. J. Van Campenhout
Promotor: prof. Dr. Ir. R. Van de Walle
Supervisors: Dr. Wesley De Neve, Ir. Tom De Nies, Joachim Van Herwegen & Ir. Miel Vander Sande

Summary

The market of digital comics is growing, however, the formats for delivering comics to its reader are mostly proprietary and have different features. The work conducted in the field of comics shows us that a wide variety of problems are currently being investigated. The goal of this thesis is to solve these problems, using the EPUB 3 standard. A solution will be proposed that focuses on the digital presentation of a comic book and the incorporating of machine-understandable metadata. The requirements for both elements and their implementation are discussed, followed by an evaluation and discussion of the conducted work.

Keywords: Digital comic, EPUB 3 format, Enhanced presentation, Linked machine-understandable metadata, Dicera
Enhanced Presentation and Machine-Understandable Metadata for Digital Comics using Open Web Formats

Pieter Heyvaert

Supervisor(s): Prof Dr. Ir. Rik Van de Walle, Ir. Miel Vander Sande, Ir. Tom De Nies, Dr. Wesley De Neve & Joachim Van Herwegen

Abstract—This paper offers a solution for the problems currently experienced in the field of digital comics. The need for extraction methods to retrieve information from the current comics, and the overhead caused by the use of different (proprietary) formats, with different features, by the publication platforms can be solved using the solution proposed in this paper. The foundation of the solution is the EPUB 3 format, where the Open Web Formats are used to fulfill the requirements of the presentation and the metadata of a digital comic. These are the support for animations, reading assistance, audio and multiple languages, together with additional administrative metadata and the introduction of descriptive metadata. In this paper, we also discuss the ideal workflow and evaluate the solution. We end with a discussion of the future work and a conclusion.

Keywords— Digital comic, EPUB 3 format, Enhanced presentation, Linked machine-understandable metadata, Dicera

I. INTRODUCTION

THE Market for digital comic books is growing, a market consisting out of people familiar with the comic series and newcomers. The revenue numbers of the past years [1] verify this statement: a growth from $640 million in 2011 to $680 million in 2012 for print comics and from $25 million in 2011 to $70 million in 2012 for digital comics. After a decline in 2010 for print comics, the revenue for these comics is increasing again, together with the increasing revenue for digital comics, as shown in Figure 1. It also proves that the growth of the market for digital comic books does not make the market for print comics shrink.

An author has a number of available distribution channels at his disposal such as comiXology, iBook Store, etc. These channels, however, use different (proprietary) formats to store and offer comics. A number of them do not make the distinction between comics and textbooks. The use of different formats results not only in storing the same information in a different package, however, a number of platforms require the author to add additional information, so that the comic is able to use every feature of the dedicated application. All this increases the overhead of the production process, while an author in the first case just wants to focus on telling stories. Platforms exist to convert a book to the correct format for the different distribution channels, however, this is cumbersome for the authors and the feature set of the formats used by those channels are not the same. Other problems, derived from related work [2][3][4][5][6][7][8][9][10][11], are based on the extraction of information from comics both in the domain of the presentation and the metadata. In a number of comics this information is available, however, not in a machine-understandable way. In the other comics no such information is available at all. We conclude that these problems are caused by the digital format in which the comic books are stored. In this paper, we propose a solution to

- circumvent the conversion of a book in the different formats required for the distribution channels, to
- evade the use of extraction methods by providing the necessary information through machine-understandable metadata, and to
- work towards a true digital comic, including the use of animations, reading assistance, audio, multi-language support and machine-understandable metadata.

II. REQUIREMENTS

The proposed solution exists out of two major elements: the presentation part and the metadata part. The requirements of the first part are

- **Req. 1** Animations,
- **Req. 2** Support different types of devices, together with
- **Req. 3** Reading assistance,
- **Req. 4** Audio, and
- **Req. 5** Multi-language support.

The requirements of the second part are
III. PROPOSED SOLUTION

A. EPUB 3

We choose the EPUB 3 format as a foundation for our solution, because it is widely used [12], the purpose of the format denotes digital publishing, hence also digital comics, and the Open Web Formats (OWF), i.e., HTML5, CSS3 and JavaScript, allow for a large range of possibilities.

B. Presentation

Before we tackle the requirements, we define the way the graphical information is stored. We do not store a page as one image, however, we store each panel separately. We take it even further: by using layering, we will store every panel, not as a single image, however, as a group of images. Every image is called a layer, hence the name layering. We can create a layer for the background, a layer for each character and a layer for each text element (e.g., speech balloon, caption and effect). This allows us to do manipulations on separate parts of the panel itself and not only on the whole panel. Listing 1 gives a basic example, using only HTML5.

Listing 1: Layering example

```
<div id="panel">  
    <div id="panel_background">  
        <img src="/background.png" alt="background image"/>
    </div>  
    <div id="panel_character">  
        <img src="/character_cell.png" alt="character cell image"/>
    </div>  
    <div id="panel_caption">  
        <img src="/caption.png" alt="caption image"/>
    </div>  
</div>
```

B.1 Animations

To support animations in our solution, we rely again on the OWF. We use JavaScript and the CSS property display to show the different layers. More advanced animations can be created, however, this is the topic of future work. An example, given in Figure 2, illustrates the concept of animations. We start with the background of a panel, followed by the characters, speech balloons and the text.

B.2 Reading Assistance

When talking about reading assistance we are talking about zooming in on the panel that the user is currently reading and navigating to the new panel once the whole panel has been read (determined by user input). To accomplish this, we make use of a jQuery\(^1\) plugin called Zoomooz\(^2\). This JavaScript library enables to zoom in onto elements of Web pages. In our case, these are the panels of a comic. An important remark is that the reading assistance accomplished here is independent of the application, because it is achieved through scripting in the EPUB file itself, and can, hence, be different for every comic book.

B.3 Audio

To add audio to the digital comic, we use the `<audio>` element of HTML5, together with JavaScript to trigger the playing of the audio, at the right moment.

B.4 Multiple Languages Support

To enable multiple languages in the same digital comic and to allow the user or application to switch between the languages, we store the different translations of a piece of text in different ‘sublayers’\(^3\). This allows to show the (text) sublayers of the currently selected language and hide the other languages. (Un)hiding the different sublayers is again accomplished by JavaScript. In Listing 2 an example can be found of the use of two languages, Dutch and English. We have one layer representing all the text, which means both languages. Inside this layer we have another `<div>` element representing the text object, needed for the addition of metadata. This `<div>` element has two `<span>` elements\(^4\), one for each language. We use the `xml:lang` attribute to denote the Dutch and English translation. If, for example, we want to read the comic in English, we set the CSS property display to none of all the `<span>` elements with the value of the `xml:lang` attribute set to nl, which makes those elements hidden, leaving only the English translation visible. Also using CSS we can set the font of the text, together with the size and the position on the panel.

Listing 2: Multi-language support example

```
<div id="panel_text">  
    <div id="panel_text_obj">  
        <span id="panel_text_nl" xml:lang="nl">How het begon ...</span>
        <span id="panel_text_en" xml:lang="en">How it began ...</span>
    </div>  
</div>
```

---

1. http://jquery.com
3. `<span>` elements instead of `<div>` elements are used
4. the sublayers previously mentioned

Fig. 2: Animation sequence
As the attentive reader will have noticed, most of the
JavaScript code is not inherit to a single comic, however, it is
reusable for a lot of comics. Here, we suggest that it is useful to
create a JavaScript library to bundle all the reusable code. First, we list the requirements of such a library. Second, we present
our own implementation, called comicreader.js.

The first part of the library should consist out of the functions
to provide the animations. Besides the standard ‘appear’ ani-
mation, other (basic) animations could also be added, such as
slide, dissolve, etc. The functions should also be designed in
a way that they can be reused if a user (here, a developer and
not the reader) wishes to create its own animations. The sec-
ond part is concerned with the reading assistance, for which we
now rely on the Zoomooz library (and jQuery). Another part of
the library handles the support for different languages and more
specific: the switching between the languages by manipulating
the different layers.

Besides listing the different functionalities the library should
offer, we already want to address one design issue: the automatic
detection of the reading order of each panel. This information
will be incorporated in the metadata, however, for the presen-
tation of the comic we also need this information. There are
two options. The metadata is used or another way of providing
this information is created. Why would we add this information
twice, and thereby creating redundancy? The information
provided by the metadata solution creates an indirect, hence
we will need to ‘parse’ more code of the XHTML pages be-
tween the different layers.

The common functionality of the pages from the EPUB file
constructed while conducting this research is grouped in a
JavaScript library comicreader.js. It does not qualify to all the
requirements stated above, however, it already shows the usefulness of developing a dedicated JavaScript library.

As can be found in the specification of EPUB 3 [13], relying
on scripting to deliver the content to its users is not allowed. It
can only be used to enhance the experience. Hence, we need to
have a fallback in case scripting is not possible or (temporarily)
disabled on a device (or in an application). Our fallback method
displays the whole page at once. We do not use layering any-
more, there are no animations, no reading assistance, no audio
and no multi-language support. We included for every panel an
image as a fallback, which will be displayed when it is not pos-
sible to execute scripts, as can be seen on Listing 3. The text on
each of those panels is hard coded, so a creator has to choose a
language in advance, it can not be altered at runtime.

As can be found in the specification of EPUB 3 [13], relying
on scripting to deliver the content to its users is not allowed. It
can only be used to enhance the experience. Hence, we need to
have a fallback in case scripting is not possible or (temporarily)
disabled on a device (or in an application). Our fallback method
displays the whole page at once. We do not use layering any-
more, there are no animations, no reading assistance, no audio
and no multi-language support. We included for every panel one
C. Metadata

The reasons to incorporate (more extensive) metadata in comic books can be found in the following list.

- Searchability
- Discoverability
- Maintainability
- Connectability

C.1 Different Types of Information

We can divide all metadata information into two groups based on the location where the information is stored: locally or remotely. With remote information we denote all the information that is not necessarily stored inside the EPUB file itself. A publisher can store all the information of a certain comic character on a server, e.g., biography, all the comics he/she appears in, etc. Ideally, the information stored in an EPUB file is not that extensive, e.g., the name of the character.

C.1.a Remote Information. Why store information remotely and not everything locally in the EPUB file? First, storing all the information would add to the size of the EPUB file. Second, this creates redundant information, i.e., what to do when there is an error in the biography, how is the EPUB file updated on the device of the user? When new comics are being published how is the information in the EPUB files, that are already downloaded, updated?

C.1.b Local Information. The information stored locally is limited to the necessary information to enable the user to gain at least the minimum advantages over having no metadata information. In what follows we will first list all the information that is present and second, we’ll give more information about each element in the list. The list is a follows.

- Character
- Location
- (Special) object
- Story arc
- Issue
- Genre
- Content rating

For every character the name (in multiple languages) can be added, we can denote whether he/she appears on the cover and the global identifier (used to request more information from, e.g., a remote server). For a location and a (special) object, the same is information available. Every comic can be associated with a certain story arc. This happens through an issue entity, which also includes the issue number. For the genres (which can be more than one for a single comic) the name (in different languages) can be given and how many percent it matches, because they can be quantified, i.e., a story can be only 25% drama (not necessarily 100%). Rating the content is done by specifying the system that is used for rating (e.g., ‘Marvel Content Rating’) and the actual rating (e.g., ‘T’, for Teen).

To structure the concepts explained in the previous section, we build an ontology using RDF. The ontology is called Dicera (the DLigital Comic book ERA vocabulary). We choose RDF because it can be used together with RDFa (more specific RDFa Core 1.1 [16]), which enables us to add metadata to Web documents. It is allowed in the new EPUB 3 (draft) specification to add RDFa tags to the XHTML pages. The use of the schemas provided by Schema.org and FOAF have been considered to be incorporated in the ontology, however, they are lacking the necessary classes and properties. For the full ontology in RDF format (and OWL format with enhanced semantics), we refer to http://semweb.mmlab.be/ns/dicera.

C.1.c Layer Unavailability. Layer unavailability puts limitations on the presentation part of our solution. The limitations on the metadata part are less severe. Each panel can still be annotated with his character, however, all characters will be included on the same layer. The same goes for the text elements and special objects. Even if the text is hard coded on the image, text objects can still be used. These objects will solely be used for metadata purposes and not for both presentation and metadata as in the normal case. It is still possible to denote the location of a panel.

The only real limitation is the following. If the metadata is used to determine all the panels where a character is present, it is only possible to get the complete panel/image. When layers are present, it is possible to retrieve images containing only the requested character, without the background, the other characters, the text elements and the special objects.

IV. Workflows & Authoring Environments

A. Ideal Workflow

The ideal workflow is the workflow that should be followed to maximally benefit the advantages of the proposed solution.

A.1 Creation of Content

During the creation of the content of the comic, which includes the creation of the pages, the panels and the cover, the creation of layers has to be taken into account. The original method of creating a comic consists out of drawing every panel with the result in mind. They do not take layering in mind, which is necessary to benefit the advantages of the solution.

A.2 Enhancement of Presentation

To enhance the presentation of the content, we use an application that relieves the authors from the burden of dealing with the technicalities of the methods discussed in Section III-B. In what follows we will discuss how the authors will use this application, hence this can be used to define the requirements if someone would consider developing such an application.

After all the layers are digitized (if not already), they are loaded in the application. We group all the layers belonging to the same panel and the panels belonging to the same page. This is followed with the addition of the text (if necessary) in multiple languages, together with the positioning of the text on the panel. After that the different animations (if any) for the different layers are defined for each panel. At this moment, also audio can be added. This all is finished with defining the reading order of the pages and the panels.
A.3 Creation of Metadata

After completing the presentation of the comic, we move on to the creation of the metadata. This can be done using the same application.

For every layer the author denotes what is present. He/she tells the application if it is dealing with a character, an object, etc. If, for example, the application is dealing with the case that it is a character, he/she connects it with the name of a character (or more specific the character object as part of the ontology, see Section III-C.1.b).

A.4 Generation of EPUB File

When both the necessary presentation and metadata information are present, it is now possible to generate an EPUB file of the comic. This can again be done using the application previously defined. The author should have the option to export the comic to the format defined in this paper. Again, the user of the application is not confronted in any way with the technicalities of our solution.

B. Authoring Environment 2.0

Authoring Environment 2.0, the authoring environment developed by Multimedia Lab5 for Boek.be6, is a proof-of-concept created to showcase a HTML5 based workflow for creating and exporting of ebooks. After using the environment, it is clear that its main focus is textbooks. The solution proposed in this paper is not incorporated in this environment, hence, it is not possible to create such a comic. An additional feature that should be considered is the addition of support for our solution. The (minimal) requirements are the same as those listed in Section IV-A. This would promote the use of our solution if this environment is being used in the field. Authors, using the platform, will be able to experiment with comics in an environment where they are already accustomed to for creating textbooks.

V. EVALUATION

A. Extraction Methods

The need for extraction methods, as mentioned in Section I, becomes superfluous, because all the required information is already available in the solution.

B. EPUB File Size

First, comparing the size of our comic to the sizes of textbooks in EBUP format shows us that the file size of our comic is 10,000 times bigger than an average textbook. Of course the main (and only) reason for this is the addition of image files in the comic.

Second, the size of the EPUB files grows if the screen resolution grows. Should a publisher then offer different files with different resolution for the images? The publisher is able to do that or he/she might opt for one of the large sizes, because the images can be downscaled if necessary. This approach, of course, leads to larger file sizes, however, zooming in on the images results in more detail.

C. Distribution Channels

The introduction (Section I) mentions the use of different (proprietary) formats by the different distribution channels causing additional work for publishers and authors to make their work available on these platforms. From the work conducted here, we conclude that the features offered by those different platforms are replicable using our proposed solution. Hence, we advice the companies behind these platforms to revise our solution, and eventually replace their current format by ours. This eases the publication of content for authors to their channel, and a standardized format is used that is actively being developed. A point of criticism might come from big publishers saying that they need their own dedicated application to offer a special and unique experience to their readers. This, however, is not valid. The presentation functionality does not depend on the reading system. It is embedded in the EPUB file itself, allowing it to be read on every system supporting the EPUB 3 specification. It also allows them to discontinue the development of their custom application for reading comics or to work towards a reading system supporting the (complete) EPUB 3 specification.

D. Reading System Support

Readium7 (with the use of scripting) and the Kobo Glo8 (using the fallback panels) allow us to read the comic. iBooks9 and Calibre10 are not able to display the comic in a correct way, both with or without scripting.

VI. FUTURE WORK

Future work has to be done both in the direction of the enhancement of the proposed solution and the creation of applications, reading systems and frameworks that work with this format to enhance its adaptation. Enhancing the solution is done by investigating the support of different types of comics, optimizing and extending the presentation features, extending the ontology, studying the use of the new (draft) specification for EPUB 3, and how it can be used to recommend new comics to readers. Work needs to be conducted to create a remote server to work with our solution and a framework that links other multimedia such as movies to comic books, and to develop a reading system that fully supports our solution.

VII. CONCLUSION

In this paper, we offer a solution for the problems studied in the literature, and the problems caused by the use of different (proprietary) formats for the different publication platforms.

The proposed solution consists out of a presentation part and metadata part, and builds on the EPUB 3 format. The presentation part offers animations, reading assistance, audio and multiple languages, through the introduction of layering and the use of Open Web Formats. The availability of scripting is an important requirement of our solution, however, when a reading system lacks this feature, a fallback procedure is foreseen. Research towards the enhancement of the EPUB 3 specification,

5 http://multimedialab.elis.ugent.be
6 http://www.boek.be
7 http://readium.org/
8 http://www.kobo.com/koboglo
9 http://www.apple.com/ibooks/
10 http://calibre-ebook.com
when it comes to comic books, has been conducted, and can be possibly used together with our solution. The metadata part enhances the administrative metadata of EPUB 3, and adds descriptive information. This is done through the use of both local information in the EPUB file and remote information on, for example, a remote server. Both types of information can be linked, which enables reading system to request more information based on the metadata available in the EPUB file. To structure the metadata, we designed an ontology called Dicera. It is available in both RDF and OWL format, allowing it to be used together with RDFa, which is recently added to the EPUB 3 specification.

The introduction of a new (or updated) format might mean a change in the workflow of creating comic books. In this paper, we discussed the ideal workflow, when using our solution. It consists out of creating the content, enhancing the presentation, creating the metadata and generating the EPUB file.

Evaluation of the proposed solution, against the problems denoted at the beginning of this section, shows us that it solves most of the problems and should be considered as the default format for distributing digital comics. However, the increasing file size, and the sometimes limited EPUB 3 support of reading systems points us to possible problems, when deploying this solution.

REFERENCES

Verbeterde Presentatie en Machinebegrijpbare Metadata voor Digitale Strips gebruikmakend van Open Web-Formaten

Pieter Heyvaert

Begeleider(s): Prof Dr. Ir. Rik Van de Walle, Ir. Miel Vander Sande, Ir. Tom De Nies, Dr. Wesley De Neve & Joachim Van Herwegen

Abstract— Deze paper biedt een oplossing voor de problemen die momenteel ervaren worden op het gebied van digitale strips. Er is momenteel nood aan extractiemethoden om informatie te halen uit de huidige strips. Er wordt overbodig werk gecreëerd door het gebruik van verschillende (gepatenteerde) formaten, met elke hun eigen eigenschappen, door de verschillende publicatieplatformen. Deze problemen kunnen opgelost worden door het werk in deze paper. De basis voor de oplossing is het EPUB 3 formaat, waar Open Web Formaten gebruikt worden om aan de presentatie- en metadataverwachtingen voor digitale strips te voldoen. Er is ondersteuning voor animaties, leeshulp, audio en meertaligheid, samen met extra administratieve metadata en de introductie van beschrijvende metadata. In deze paper bespreken we ook de ideale workflow en wordt de oplossing geëvalueerd. We eindigen met de bespreking van toekomstig werk en een conclusie.

Trefwoorden— Digitale strip, EPUB 3 formaat, Verbeterde presentatie, Gelinkte machinebegrijpbare metadata, Dicera

I. INTRODUCTIE


Een auteur heeft verschillende distributiekanalen ter beschikking waaronder comiXology, iBook Store, etc. Deze kanalen gebruiken verschillende formaten om de strips op te slaan en aan te bieden. Verschillende onder hen maken niet het onder- scheid tussen strips en boeken. Het gebruik van de verschillende (gepatenteerde) formaten resulteert niet alleen in het opslaan van dezelfde informatie op verschillende manieren, maar zorgt er ook voor dat de auteur extra informatie moet toevoegen. Deze informatie is nodig zodat de strip volop gebruik kan maken van de voordelen van de gebruikte applicaties. Dit zorgt voor een verhoogde werklast, terwijl de auteur zich in hoofdzaak wil bezighouden met het vertellen van verhalen. Er bestaan platformen om een boek om te zetten naar het correcte formaat voor de verschillende kanalen, maar dit is hinderlijk voor de auteur en de eigenschappen van de verschillende formaten zijn niet hetzelfde. Andere problemen, zoals gevonden in [2][3][4][5][6][7][8][9][10][11], zijn gebaseerd op het extraheeren van informatie van strips, zowel in het domein van de presentatie als van de metadata. In een aantal strips is deze informatie aanwezig maar niet op een machineter- staanbare manier. In andere strips is deze informatie totaal afwezig. We kunnen besluiten dat deze problemen veroorzaakt worden door het digitale formaat waarin de strips zijn opgeslagen. In deze paper stellen we een oplossing voor om

- het omzetten van een boek in verschillende formaten, nodig voor de distributiekanalen, te vermijden, om
- het gebruik van extractiemethoden te vermijden door de nodige informatie te voorzien op een machine-verstaanbare manier, en om
- te werken in de richting van een echte digitale strip, voorzien van animaties, leeshulp, audio, meertaligheid and machine-verstaanbare metadata.

II. VEREISTEN

De voorgestelde oplossing bestaat uit twee grote onderdelen: het presentatie deel en het metadata deel. De vereisten voor het eerste deel zijn

Vw. 1 Animaties,
Vw. 2 Ondersteuning voor verschillende types van apparaten, samen met
Vw. 3 Leeshulp,
Vw. 4 Audio, en
Vw. 5 Meertaligheid.

De vereisten voor het tweede deel zijn

Vw. 1 Het uitbreiden van de administratieve metadata,

en

Vw. 2 De ondersteuning voor beschrijvende metadata.

III. VOORGESTELDE OPLOSSING

A. EPUB 3

We hebben gekozen voor het EPUB 3 formaat als basis voor onze oplossing, omdat het wereldwijd gebruikt wordt [12] en omdat het doel van dit formaat digitaal publiceren is, waaronder dus digitale strips. Het formaat maakt ook gebruik van Open Web Formaten (OWF), zoals HTML5, CSS3 en JavaScript, welke een groot aanbod aan mogelijkheden bieden.

B. Presentatie

Voor we de vereisten behandelen, definieren we eerst de manier waarop de grafische informatie wordt opgeslagen. We slaan niet elke pagina op als één afbeelding, maar we slaan elk paneel apart op. We trekken deze lijn door: door gebruik te maken van lagen, slaan we elk paneel niet op als één afbeelding, maar als een groep van afbeeldingen. Elke afbeelding wordt een laag genoemd. We kunnen een laag creëren voor de achtergrond, voor elk personage en voor elk tekstelement (bv. tekstballonnen, opschriften en effecten). Dit laat ons toe om manipulaties te doen van de verschillende onderdelen van een paneel en niet alleen van het volledige paneel. Listing 1 geeft een simpel voorbeeld, waarbij enkel HTML5 gebruikt wordt.

Listing 1: Lagen voorbeeld

```html
<div id="panel">
  <div id="panel_background">
    <img src="background.png" alt="background image"/>
  </div>
  <div id="panel_character">
    <img src="character.png" alt="character cell image"/>
  </div>
  <div id="panel_caption">
    <img src="caption.png" alt="caption image"/>
  </div>
</div>
```

B.1 Animaties

Om animaties te ondersteunen in onze oplossing, maken we gebruik van OWF. We gebruiken JavaScript en de CSS eigenschap display om de verschillende lagen te tonen. Meer gevanceerde animaties kunnen ontworpen worden, maar dit is onderwerp voor toekomstig werk. Een voorbeeld is gegeven in Figuur 2 en illustreert het concept van animaties. We starten met de achtergrond van het paneel, gevolgd door de personages, tekstballonnen en de tekst.

B.2 Leeshulp

Wanneer we praten over leeshulp, praten we over het inzoomen op een paneel dat de gebruiker op dat moment aan het lezen is en het navigeren naar een nieuw paneel wanneer het huidige paneel volledig gelezen is (bepaald door invoer van de gebruiker). Om dit te verwezenlijken, maken we gebruik van een jQuery\(^1\) plugin genaamd Zoomooz\(^2\). Deze JavaScript bibliotheek laat ons toe om in te zomen op elementen van een Webpagina. In ons geval zijn dit de panelen van de strip. Een belangrijke opmerking hier is dat leeshulp verwezenlijkt wordt onafhankelijk van de gebruikte applicatie, aangezien scripting in het EPUB bestand zelf gebeurt. Deze kan dan ook verschillend zijn voor elke strip.

B.3 Audio

Om audio toe te voegen aan digitale strips, maken we gebruik van het <audio> element van HTML5. Dit kan samen met JavaScript gebruikt worden om het op de juiste moment af te laten afspelen.

B.4 Meertaligheid

We willen verschillende talen toelaten in dezelfde strip en we willen de gebruiker of de applicatie toelaten om tussen de verschillende talen te wisselen. Dit doen we door de verschillende vertalingen op te slaan als ‘sublagen’\(^3\). Dit laat toe om de (tekst)sublaag van de huidige taal te tonen en de andere talen te verbergen. Het tonen en verbergen van sublagen gebeurt opnieuw via JavaScript. In Listing 2 kan een voorbeeld gevonden worden waarbij twee talen gebruikt worden, nl. Nederlands en Engels. We hebben één laag die de twee talen bevat. Hierin hebben we nog een <div> element dat een tekstobject voorstelt, nodig voor het toevoegen van metadata. Dit <div> element heeft twee <span> elementen\(^4\), één voor elke taal. We gebruiken het xml:lang attribuut om te zeggen over welke vertaling

1http://jquery.com
2http://jaukia.github.io/zoomooz/
3<span> elementen in plaats van <div> elementen worden gebruikt
4de sublagen eerder vermeld
het gaat. Als we bijvoorbeeld de strip in het Engels willen lezen dan zetten we de CSS eigenschap display op none voor alle span elementen met de waarde van hun xml:lang attribuut gelijk aan nl. Dit zorgt ervoor dat deze elementen verborgen worden zodat enkel de Engelse vertaling nog zichtbaar blijft. Ook gebruikmakende van CSS kunnen we het lettertype, de grootte en de positie op het paneel van de tekst bepalen.

Listing 2: Meertaligheid voorbeeld

```html
<div id="panel_text">  
  <div id="panel_text_obj">  
    <span id="panel_text_nl" xml:lang="nl">Hoe het begon</span> 
    How it began ... </span> 
    <span id="panel_text_en" xml:lang="en"></span> 
    How it began ... </span> 
  </div> 
</div>
```

B.5 Gebruik van Scriptingtalen

De oplettende lezer zal bemerkt hebben dat de meeste JavaScript code niet enkel toepasbaar is op één strip, maar dat deze herbruikbaar is voor andere strips. Daarom stellen we voor om een JavaScript bibliotheek te maken die alle herbruikbare code bevat. Eerst lijsten we alle vereisten op van zo een bibliotheek. Daarna stellen we onze eigen implementatie voor, genaamd comicreader.js.

Het eerste deel van de bibliotheek moet bestaan uit de functies die nodig zijn om animaties te voorzien. Naast de standaard ‘tonen’ animatie, kunnen ook andere animaties voorzien worden, nl. ‘schuiven’, ‘oplossen’, etc. Deze functies moeten zo ontwikkeld worden dat deze hergebruikt kunnen worden door de gebruiker (hier betreft het de ontwikkelaar en niet de lezer) indien deze eigen animaties wil creëren. Het tweede deel houdt zich bezig met de leeshulp, voor welke we nu vertrouwen op de Zoomoz bibliotheek (en jQuery). Een ander deel van de bibliotheek zorgt voor de meertaligheid en het wisselen tussen twee talen door het manipuleren van de lagen.

Naast het oplijsten van de functionaliteit die de bibliotheek moet aanbieden, willen we ook al een ontwerpsprobleem aanpakken: het automatisch detecteren van de leesvolgorde voor elke paneel. Deze informatie wordt in de meta-data, maar deze informatie is ook nodig voor de presentatie. Er zijn twee mogelijkheden. De metadata wordt gebruikt of er wordt een andere manier gebruikt om deze informatie te voorzien. Waarom zouden we deze informatie tweemaal opslaan en hierdoor redundantie creëren? De informatie voorzien door de metadata zorgt voor een indirectie, want meer code zal er bekeken moeten worden voor de informatie verkregen worden. Wat resulteert in een vertraging. De vraag (en wat dus moet onderzocht worden) is of dit enig verschil maakt: zorgt de indirectie voor een grote vertraging of niet. Als er een vertraging ontstaat kan deze omzeild worden door alle informatie op voorhand in te lezen als het EPUB bestand geladen wordt. Dit zorgt voor een grotere laadtijd, maar er zal geen vertraging zijn bij het lezen van de strip. Als de laadtijd of de vertraging een probleem vormt, kunnen we een data-readingorder attribuut toevoegen aan de div tag van een paneel, welke ons toelaat om deze indirectie te omzeilen en de performantie te verhogen.

De gemeenschappelijke functionaliteit van de pagina’s van het EPUB bestand, gemaakt tijdens dit onderzoek, is gegroepeerd in de Java Script bibliotheek comicreader.js. Het voldoet niet aan alle vereisten die hierboven vermeld werden, maar het toont het nut van het ontwikkelen van een gespecialiseerde JavaScript bibliotheek.

B.6 Terugvalmethode Wanneer Scripting Niet Beschikbaar

Zoals terug te vinden is in de specificatie van EPUB 3 [13] mogen we niet betrouwen op scripting om inhoud af te leveren aan de gebruiker. Het mag enkel gebruikt worden om de gebruikerservaring te verbeteren. Dit wil dus zeggen dat we een terugvalmethode moeten voorzien indien scripting niet mogelijk is of (tijdelijk) uitgeschakeld is op het apparaat (of in de applicatie). Onze terugvalmethode toont de volledige pagina in één keer. We gebruiken geen lagen meer, er zijn geen animaties, er is geen leeshulp, geen audio en geen meertaligheid. We voorzien voor elk paneel één afbeelding om op terug te vallen. Deze wordt getoond indien het niet mogelijk is om scripts uit te voeren, zoals te zien in Listing 3. De tekst op elk van deze panelen is hard gecodeerd, m.a.w. een auteur moet dus op voorhand kiezen in welke taal dit zal zijn, want deze kan niet meer aangepast worden tijdens het lezen.

Listing 3: Terugvalmethode voorbeeld

```html
<div id="panel_fallback" class="panel_fallback"> 
  <img src="fallback.png" alt="panel fallback image"/> 
</div>
```

B.7 Onbeschikbaarheid van Lagen

In de vorige secties gingen we er steeds van uit dat een strip (of meer specifiek elk paneel) beschikbaar is in lagen. Wat gebeurt er als dit niet het geval is of als oudere strips (zonder lagen) omgezet dienen te worden in het formaat voorgesteld in deze paper? In dit geval, nemen we aan dat elk paneel voorgesteld wordt door één afbeelding.

Animaties toevoegen aan een paneel is niet mogelijk, aangezien de nodige lagen niet beschikbaar zijn. Meertaligheid is ook niet mogelijk aangezien de tekst hard gecodeerd is op de afbeelding. Audio kan nog steeds toegevoegd worden, net als leeshulp.
B.8 EPUB Regio-gebaseerde Navigatie en Meerdere-vertolking Publicaties


C. Metadata

De reden om (uitgebreidere) metadata te voorzien voor strips kan gevonden worden in de volgende lijst.

- Zoekbaarheid
- Ontdekkbaarheid
- Onderhoudbaarheid
- Verbindbaarheid

C.1 Verschillende Types van Informatie

We kunnen de metadata informatie opdelen in twee groepen gebaseerd op de locatie waar de informatie opgeslagen wordt. Dit kan lokaal of op afstand zijn. Met informatie op afstand bedoelen we alle informatie die niet noodzakelijk opgeslagen wordt in het EPUB bestand zelf. Een uitgever kan ervoor kiezen om bepaalde informatie over een personage op te slaan op een server, zoals de biografie, alle strips waar hij/zij in voor- komt, etc. Idealiter is de informatie opgeslagen in het EPUB bestand niet te uitgebreid, bijvoorbeeld enkel de naam van het personage.

C.1.a Informatie op Afstand. Waarom zouden we informatie op afstand opslaan en niet alles lokaal in het EPUB bestand? Eén, alle informatie opslaan zou de grootte van het EPUB bestand laten toenemen. Twee, het zorgt voor redundantie. Wat dient er bijvoorbeeld te geheuren als er een fout in de biografie is? Hoe kan het EPUB bestand aangepast worden dat reeds op het apparaat van de gebruiker staat? Wanneer nieuwe strips uitgebracht worden, hoe wordt de informatie in de reeds gepubliceerde (en gedownload) strips dan aangepast?

C.1.b Locale Informatie. De informatie die lokaal opgeslagen wordt, is gelimiteerd tot de nodige informatie zodat de gebruiker minstens enig voordeel ondervindt t.o.v. het niet beschikken over metadata. In wat volgt zullen we alle beschikbaar informatie oplijsten. Dit wordt gevolgd door meer informatie over elk element van de lijst. De lijst is al volg.

- Personage
- Locatie
- (Speciaal) voorwerp
- Verhaalboog
- Uitgave
- Genre
- Beoordeling

Voor elk personage kan de naam (in meerdere talen) toegevoegd worden. We kunnen aanduiden of hij/zij voorkomt op de coverpagina en we kunnen een globale identificatie meegegeven. Deze identificatie wordt gebruikt om meer informatie op te halen van bijvoorbeeld een server. Voor een locatie en een (speciaal) voorwerp is dezelfde informatie beschikbaar. Elke strip kan geassocieerd worden met en bepaalde verhaalboog. Dit be- luurt door een uitgave entiteit, welke ook het uitgavenummer bevat. Voor de genres (dit kunner er meer dan één zijn per strip) is het mogelijk de naam (in verschillende talen) te geven, samen met het aantal percent dat het overeenkomt. We doen dit omdat een genre gekwantificeerd kan worden, bijvoorbeeld, een ver- haal kan slechts 25% drama zijn (en niet noodzakelijk 100%). Het beoordelen van de inhoud gebeurt door het systeem aan te duiden dat gebruikt wordt (bijvoorbeeld ‘Marvel Content Rating’) en de beoordeling zelf (bijvoorbeeld, ‘T’, voor tiener).

Om de verschillende concepten, van de vorige sectie, te structuren hebben we een ontologie gebouwd, gebruikmakend van RDF. De ontologie heet Dicera (the Digital Comic book ERA vocabulary). We kozen voor RDF omdat het gebruikt kan worden samen met RDFa (meer specifiek RDFa Core 1.1 [16]), wat ons toelaat om metadata toe te voegen aan Webdocumen- ten. Het is toegelaten door de nieuwe EPUB 3 (draft) specificatie om RDFa tags toe te voegen aan XHTML pagina’s. Het gebruik van de schema’s voorzien door Schema.org en FOAF werd overwogen, maar deze blijken niet de nodige klassen en relaties te bevatten. Voor de volledige ontologie in RDF formaat (en OWL formaat met verbeterde semantiek), verwijzen we naar http://semweb.mmlab.be/ns/dicera.


De enige echte beperking is de volgende. Als metadata gebruikt wordt om te bepalen op welke panelen een bepaald personage voorkomt, dan is het enkel mogelijk om het volledige paneel/afbeelding op te vragen. Als lagen aanwezig zijn dan is het mogelijk om afbeeldingen op te vragen waar enkel het personage opstaat, zonder de achtergrond, de andere personages, de tekstelementen en de speciale voorwerpen.
IV. WORKFLOWS & AUTHEURSOMGEVINGEN

A. Ideale Workflow

De ideale workflow is de workflow die gevolgd dient te worden om maximaal van de voordelen van deze oplossing te genieten.

A.1 Creatie van Inhoud

Tijdens de creatie van de inhoud van de strip, moet er rekening gehouden worden met de creatie van lagen. De originele methode die gebruikt wordt, bestaat uit het tekenen van elk paneel met het resultaat voor ogen. Lagen worden niet in rekening gehouden. Deze zijn natuurlijk nodig om te genieten van de voordelen van onze oplossing.

A.2 Verbetering van Presentatie

Om de presentatie van de inhoud te verbeteren, gebruiken we een applicatie dat de auteur verlost van de technische elementen van de methoden besproken in Sectie III-B. In wat volgt zullen we bespreken hoe de auteur deze applicatie zal gebruiken. Met andere woorden, dit kan gebruikt worden om de vereisten van zuks programma op te stellen, indien het ontworpen wordt.

Nadat alle lagen gedigitaliseerd zijn (indien nodig), worden ze ingeladen in het programma. Alle lagen die tot hetzelfde paneel behoren worden gegroepeerd, net als de panelen die bij dezelfde pagina horen. Dit wordt gevolgd door het toevoegen van tekst (indien nodig) in meerdere talen, samen met het positioneren op het paneel. Hierna worden de verschillende animaties (indien nodig) voorzien voor de verschillende lagen van elk paneel. Op dit moment kan ook audio toegevoegd worden. Als laatste wordt de leesvolgorde van de pagina’s en panelen bepaald.

A.3 Creatie van Metadata

Na het afwerken van de presentatie van de strip, gaan we verder met het creëren van de metadata. Dit kan gebeuren met dezelfde applicatie.

Voor elke laag geeft de auteur aan wat erop aanwezig is. Hij/zij vertelt de applicatie of het te maken heeft met een personage, voorwerp, etc. Indien, bijvoorbeeld, de applicatie te maken heeft met een personage dan zal hij/zij deze verbinden met de naam van het personage (of specifieker het personage object, als onderdeel van de ontologie, zie Sectie III-C.1.b)

A.4 Generatie van EPUB Bestand

Wanneer zowel de presentatie en de metadata informatie aanwezig zijn, is het mogelijk om het EPUB bestand voor de strip te genereren. Dit kan opnieuw gebeuren met de eerdere gedefinieerde applicatie. De auteur moet de optie hebben om de strip te exporteren naar het formaat voorgesteld in deze paper. Opnieuw, de gebruiker van de applicatie wordt op geen enkele manier blootgesteld aan de technische kant van onze oplossing.

B. Authoring Environment 2.0

Authoring Environment 2.0, de auteursomgeving ontwikkeld door Multimedia Lab
5 voor Boek.be
6, is een proof-of-concept gecreëerd om HTML5-gebaseerd workflow voor te stellen voor het creëren en exporteren van ebooks. Na deze omgeving gebruikt te hebben, is het duidelijk dat de focus hier op tekstboeken ligt. De voorgestelde oplossing is niet verwerkt in de omgeving, wat het dus niet mogelijk maakt om zo een strip te maken. Het toevoegen van ondersteuning voor onze oplossing zou overwogen moeten worden. De (minimale) vereisten zijn dezelfde als opgelijst in Sectie IV-A. Dit promoot het gebruik van onze oplossing in een omgeving die in de praktijk gebruikt wordt. Auteurs, die gebruik maken van het platform, kunnen experimenteren met strips in een omgeving die ze al gewoon zijn om te gebruiken voor tekstboeken.

V. EVALUATIE

A. Extractie Methoden

De nood voor extractiemethoden, zoals vermeld in Sectie I, worden overbodig, aangezien alle nodige informatie aanwezig is in de oplossing.

B. EPUB bestandsgroottes

Het vergelijken van de bestandsgroottes van onze strip met deze van tekstboeken in EPUB formaat, toont aan dat de bestandsgroote 10.000 maal groter is dan deze van een gemiddeld tekstboek. De hoofd- en enige reden is het toevoegen van de afbeeldingen aan de strip.

De grootte van EPUB bestanden groeit indien de schermresolutie groeit. Moet een uitgever dan verschillende bestanden aanbieden voor verschillende resoluties voor de afbeeldingen? De uitgever heeft de optie om dat te doen of hij/zij kan er ook voor kiezen om één van de grotere formaten te kiezen, omdat afbeeldingen geschapeld kunnen worden. Deze aanpak leidt uiteraard tot grotere bestandsgroottes, maar inzoomen op een afbeelding resulteert dan weer in meer detail.

C. Distributiekanalen

De introductie (Sectie I) vermeldt dat het gebruik van verschillende (gepatenteerde) formaten door de verschillende distributiekanalen zorgt voor extra werk voor uitgevers en auteurs die hun werk op al deze platformen beschikbaar willen maken. Uit het werk van deze paper kunnen we besluiten dat de functionaliteit van de verschillende platformen verwezenlijkt kan worden door de voorgestelde oplossing. We raden de bedrijven achter deze kanalen dan ook aan om onze oplossing te overwegen en hun huidig formaat uiteindelijk te vervangen door ons formaat. Dit vergemakkelijkt de publicatie voor auteurs op hun kanaal. Er wordt dan een standaard formaat gebruikt die actief ontwikkeld wordt. Een punt van kritiek, komende van grote uitgevers, zou kunnen zijn dat ze hun eigen applicatie nodig hebben om de gebruiker een speciale en unieke ervaring te bieden. Dit is niet correct. De presentatiefunctionaliteit hangt niet af van het leesysteem. Deze zit in het EPUB bestand zelf, wat het toegelaat om gelezen te worden op elk systeem dat de EPUB 3 specificatie ondersteunt. Het laat ook toe om de ontwikkeling van hun eigen applicatie te staken of om een applicatie te ontwikkelen die de (volledige) EPUB 3 specificatie ondersteunt.

---

5 http://multimedialab.elis.ugent.be
6 http://www.boek.be
De evaluatie van de voorgestelde oplossing, tegenover de bestaande problemen vermeld in het begin van deze sectie, toont aan dat het de meeste problemen oplost. Het zou dus beschouwd moeten worden als het standaard formaat om digitale strips te verdelen. Hierbij moet wel rekening gehouden worden met de toenemende bestandsgrootte, en de soms gelimiteerde ondersteuning van leessystemen. Wat mogelijks tot problemen kan leiden bij het gebruiken van deze oplossing.

VI. TOEKOMSTIG WERK

Toekomstig werk moet uitgevoerd worden in de richting van het optimaliseren van de voorgestelde oplossing en het creëren van applicaties, leessystemen en raamwerken die gebruik maken van het formaat, om de adaptatie te verhogen. Het verber- teren van de oplossing gebeurt door onderzoek te doen naar de ondersteuning van verschillende types van strips, optimalisering en uitbreiding van de presentatie-eigenschappen, de uitbreiding van de ontologie, het bestuderen van het gebruik van de nieuwe (draft) specificatie voor EPUB 3, en hoe het kan gebruikt wor- den om nieuwe strips aan te bevelen aan lezers. Er moet werk uitgevoerd worden om een server te creëren die werkt met onze oplossing, een raamwerk dat andere multimedia zoals films linkt met strips, en om een leessysteem te ontwikkelen dat onze op- lossing volledig ondersteunt.

VII. CONCLUSIE

In deze paper bieden we een oplossing voor de problemen van de literatuur en de problemen veroorzaakt door het gebruiken van verschillende (gepatenteerde) formaten voor de verschil- lende distributiekanalen.

De voorgestelde oplossing bestaat uit een presentatie gedeelte en metadata gedeelte. Het maakt gebruik van het EPUB 3 for- maat. Het presentatie gedeelte biedt animaties, leeshulp, au- dio en meertaligheid, door het gebruik van lagen en Open Web Formaten. De beschikbaarheid van scripting is een belangrijke vereiste voor onze oplossing, maar het is ook mogelijk om een strip te lezen indien het leessysteem geen scripting voorziet. Het metadata gedeelte breidt de administratieve metadata van EPUB 3 uit en voegt beschrijvende informatie toe. Dit gebeurt door gebruik te maken van lokale informatie in het EPUB bestand en informatie op afstand, bijvoorbeeld op een server. Beide ty- pes van informatie kunnen gelinkt worden, wat ervoor zorgt dat leessystemen verzoeken kunnen doen om meer informatie te verkrijgen, gebaseerd op de informatie aanwezig in het EPUB bestand. Om de metadata te structuren hebben we een ontol- ogie ontwikkeld genaamd Dicera. Het is beschikbaar in zowel RDF als OWL formaat, wat het toelaat om gegeven toegevoegd te zijn aan de EPUB 3 specificatie.

De introductie van een nieuwe (of aangepast) formaat kan er- voor zorgen dat de workflow voor het creëren van strips aangepast moet worden. In deze paper bespreken we de ideale work- flow wanneer onze oplossing gebruikt wordt. Deze bestaat uit de creatie van de inhoud, de verbetering van de presentatie, de creatie van de metadata en de generatie van het EPUB bestand.
Contents

1 Introduction
   1.1 Problem Statement ........................................... 3
   1.2 Summary .................................................... 4

2 Literature
   2.1 Adaptation .................................................. 5
   2.2 Visualization ............................................... 6
   2.3 Metadata .................................................. 8
   2.4 Accessibility .............................................. 9
   2.5 Formats ..................................................... 10
   2.6 Conclusion ................................................ 12

3 Proposed Solution
   3.1 Requirements ............................................... 15
   3.2 Design Choices ............................................ 16
      3.2.1 Rationale for EPUB Usage ......................... 16
      3.2.2 Localization of Metadata ....................... 17
      3.2.3 Embedded and Inline Metadata ............... 17
   3.3 Solution Overview ....................................... 17

4 Presentation
   4.1 Requirements .............................................. 21
   4.2 Layering .................................................... 22
   4.3 Animations ................................................ 23
   4.4 Reading Assistance ...................................... 24
   4.5 Audio ....................................................... 28
   4.6 Multiple Languages Support ............................ 28
   4.7 Use of Scripting Languages ............................. 29
   4.8 Fallback When Scripting Is Disabled ............... 30
4.9 Layer Unavailability ............................................ 32
4.10 EPUB Region-based Navigation Specification ................. 32
4.11 EPUB Multiple-Rendition Publications .......................... 33
4.12 Summary ....................................................... 34

5 Metadata ......................................................... 35
5.1 The Need for Metadata Information ............................... 35
5.2 Different Types of Information .................................. 36
  5.2.1 Remote Information ....................................... 36
  5.2.2 Local Information ......................................... 37
5.3 Connecting Local and Remote Information ....................... 48
5.4 Layer Unavailability ............................................ 48
5.5 Summary ....................................................... 49

6 Implementation .................................................. 51
6.1 Presentation ...................................................... 51
6.2 Metadata ........................................................ 52
  6.2.1 Character-Driven Comic Search Engine with Geographical Visualization ........................................ 52
  6.2.2 Character-Driven Panel Extractor for Comics .................. 53
  6.2.3 Comic Information Viewer .................................. 55
6.3 Summary ....................................................... 60

7 Workflows & Authoring Environments .............................. 61
7.1 Ideal Workflow .................................................. 61
  7.1.1 Creation of Content ....................................... 61
  7.1.2 Enhancement of Presentation ................................. 62
  7.1.3 Creation of Metadata ....................................... 63
  7.1.4 Generation of EPUB File .................................... 63
7.2 Transition ....................................................... 64
7.3 Converting Old Comics .......................................... 64
7.4 Authoring Environment 2.0 ...................................... 64
7.5 Summary ....................................................... 65
D List of EPUB Books to Determine Average Size 109

Bibliography 113

List of Figures 117
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACBF</td>
<td>Advanced Comic Book Format</td>
</tr>
<tr>
<td>CBA</td>
<td>Comic Book Archive</td>
</tr>
<tr>
<td>CMML</td>
<td>Comic Book Markup Language</td>
</tr>
<tr>
<td>ComicsML</td>
<td>Comics Markup Language</td>
</tr>
<tr>
<td>ComicIV</td>
<td>Comic Information Viewer</td>
</tr>
<tr>
<td>CORVIS</td>
<td>COnics Rendering system on VIdeo Streaming</td>
</tr>
<tr>
<td>Dicera</td>
<td>Digital Comic book ERA vocabulary</td>
</tr>
<tr>
<td>DITA</td>
<td>Darwin Information Typing Architecture</td>
</tr>
<tr>
<td>DRM</td>
<td>Digital Rights Management</td>
</tr>
<tr>
<td>ECF</td>
<td>E-Comic Format</td>
</tr>
<tr>
<td>EPUB</td>
<td>Electronic Publishing</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>IBA</td>
<td>iBooks Author Format</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>JPEG</td>
<td>Joint Photographic Expert Group</td>
</tr>
<tr>
<td>MathML</td>
<td>Mathematical Markup Language</td>
</tr>
<tr>
<td>MMF</td>
<td>Manga Metadata Framework</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Experts Group</td>
</tr>
<tr>
<td>OGP</td>
<td>Open Graph Protocol</td>
</tr>
<tr>
<td>OWF</td>
<td>Open Web Format</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>PDF</td>
<td>Portable Document Format</td>
</tr>
<tr>
<td>PNG</td>
<td>Portable Network Graphics</td>
</tr>
<tr>
<td>RAR</td>
<td>Roshal ARchive</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>SVG</td>
<td>Scalable Vector Graphics</td>
</tr>
<tr>
<td>TEI</td>
<td>Text Encoding Initiative</td>
</tr>
<tr>
<td>XHTML</td>
<td>Extensible Hypertext Markup Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The market for digital comic books is growing, a market consisting of people familiar with the comic series and newcomers. At the end of September of 2013, 12 months after reaching 100 million downloads, comiXology\(^1\) hit 200 million downloads [18]. ComiXology, recently acquired by Amazon\(^2\) [1], is a cloud-based platform that offers digital comic books across different devices. Alongside comiXology, also Marvel Comics\(^3\), DC Comics\(^4\) and Dark Horse Comics\(^5\) have their own platform. At the end of 2012, DC Comics saw that its digital sales had gone up by 200% [26], compared to the same period the year before. This results in the fact that the total amount of downloads across all platforms will be bigger than 200 million.

A possible result of the growth in the market of digital comics is that revenue of print comics shrinks. However, this is not the case as stated by Mark Waid of Thrillbent\(^6\). He said in a panel discussion that the impact of digital comics will be neutral in the worst case, however, all evidence seems to show that it is helping [37]. Not only the opinion of one man, however, also the revenue numbers of the past years [3] verify this statement: the comic market is growing, from $640 million in 2011 to $680 million in 2012 for print comics and from $25 million in 2011 to $70 million in 2012 for digital comics. After a decline in 2010 for print comics, the revenue for these comics is increasing again, together with the increasing revenue for digital comics, as shown in Figure 1.1.

Not alone the comics themselves, however, also a number of movies based on comic series tend to bring in a steady amount of money. For example, ‘The Avengers’\(^7\) made $200 million during its opening weekend [32]. The total revenue for this movie worldwide

\(^1\)https://www.comixology.com
\(^2\)http://www.amazon.com
\(^3\)http://marvel.com
\(^4\)http://www.dccomics.com
\(^5\)http://www.darkhorse.com
\(^6\)http://thrillbent.com
\(^7\)http://marvel.com/avengers_movie/
is more than $1 billion [8]. Besides ‘The Avengers’, ‘Iron Man 3’\textsuperscript{8} made more than $1 billion worldwide [6] and ‘Spider-Man’\textsuperscript{9} $822 million [9]. From the above sales, we can conclude that comics are a multimedia branch not to be neglected, both from a revenue and readers/viewers perspective.

When it comes to the distribution of music, a customer is able to purchase his/her songs and albums via different platforms, e.g., iTunes Store\textsuperscript{10}, Amazon MP3\textsuperscript{11}. There is a similar situation for comics: it is possible to buy the newest issue of Superman through comiXology or DC Comics. Each comic is accessible through its dedicated app. Different prices and different app features may persuade the reader to purchase his/her comics through a specific platform.

The e-Strips\textsuperscript{12} project, by several research departments from UGent\textsuperscript{13} and VUB\textsuperscript{14} such

\begin{figure}
\centering
\includegraphics[width=\textwidth]{comic-book-market-revenue.png}
\caption{Comic book market revenue}
\end{figure}

\textsuperscript{8}http://marvel.com/movies/movie/176/iron_man_3
\textsuperscript{9}http://marvel.com/movies/movie/6/spider-man
\textsuperscript{10}http://www.apple.com/itunes/features/index.html#store
\textsuperscript{11}http://www.amazon.com/MP3-Music-Download/b/ref=sa_menu_mp3_str?ie=UTF8&node=163856011
\textsuperscript{12}http://www.iminds.be/en/research/overview-projects/p/detail/e-strips
\textsuperscript{13}http://www.ugent.be
\textsuperscript{14}http://www.vub.ac.be/en/
as Multimedia Lab\textsuperscript{15} and iMinds\textsuperscript{16}, aims to build a platform for all the activities from creating comics to selling them. They work together with a number of the biggest names in comics and publishing in Belgium such as WPG Uitgevers België\textsuperscript{17} and Ballon Media\textsuperscript{18}.

With the digitization of comics, new possible features are being contrived that were not or less possible with the print versions. In the next section, we will discuss what the main problem is with the current (format of) digital comics and how we will try to resolve this.

1.1 Problem Statement

An author might want to publish comics through every available channel to reach as many (new) readers as possible. The platforms discussed above use different (proprietary) formats to store comics. This is not only the same information in a different package, however, a number of platforms require the author to add additional information, so that the comic is able to use every feature of the dedicated application. All this increases the overhead of the production process, while an author in the first case just wants to focus on telling stories. Platforms exist to convert a book to the correct format for the different distribution channels, however, this is cumbersome for the authors and the feature set of the formats used by those channels are not the same. Other problems, derived from the literature, discussed in Chapter 2, are based on the extraction of information from comics both in the domain of the presentation and the metadata. In a number of comics this information is available, however, not in a machine-understandable way. In the other comics no such information is available at all. We conclude that these problems are caused by the digital format in which the comic books are stored. In this thesis, we propose a solution to

- circumvent the conversion of a book in the different formats required for the distribution channels, to
- evade the use of extraction methods by providing the necessary information through machine-understandable metadata, and to
- work towards a true digital comic, including the use of animations, reading assistance, audio, multi-language support and machine-understandable metadata.

\textsuperscript{15}http://multimedialab.elis.ugent.be
\textsuperscript{16}http://www.iminds.be/en
\textsuperscript{17}http://www.wpg.be
\textsuperscript{18}http://www.ballonmedia.be
1.2 Summary

Before presenting the work conducted in this thesis, we look at the literature around the topic of (digital) comics in Chapter 2. In Chapter 3 we discuss the proposed solution by listing the requirements, explaining the most important design choices and giving an overview of the solution. In the Chapters 4 and 5 we will discuss the two major parts of our solution: the presentation part and the metadata part, respectively. After explaining the methodologies used to fulfill the requirements of the two aforementioned elements, we give a number of more elaborate examples in Chapter 6. Offering a (new) solution for publishing digital comics might lead to the adjustment of the workflow of authors and publishers, this is the topic of Chapter 7. There, we will also discuss the authoring environment of the e-Strips project. In Chapter 8 we evaluate the proposed solution. In Chapter 9 we present the future work that needs to be conducted and we end with a conclusion about the work conducted in this thesis in Chapter 10.
Chapter 2

Literature

This chapter summarizes the work that has been conducted regarding (digital) comics. We defined four categories to keep an overview of the different topics that are being addressed in the research efforts: adaptation, visualization, metadata and accessibility. They are the topic of Sections 2.1, 2.2, 2.3 and 2.4. Section 2.5 summarizes the (proprietary) formats developed to visualize comics and/or represent the metadata information of comics. Section 2.6 draws conclusions from the discussed work.

2.1 Adaptation

Starting from a printed comic or a blockbuster movie, several research efforts have been conducted towards the adaptation of content: from automatic comic content adaptation to the conversion between movies and comics.

In [12], a content adaptation system for comics is proposed, with special attention for mobile devices. Based on existing comic book content, this system is able to create an adaptation for mobile devices to enhance the reader experience, making use of the capabilities of those devices. The authors propose a new method that allows for the automatic extraction of frames and their content, which includes text balloons and their text, resulting in better accuracy and processing time, compared to other methods. The adaptation system outputs Comic XML files for data and combines this with the XHTML mobile profile\textsuperscript{1} for presentation. The Comic XML file is based on ComicsML version 0.2 by Jason McIntosh (more information about this format later). It enhances ComicsML with layout information about comic frames and balloons (including the text inside these balloons). The strategies for the adaption of the content also take into account the speed between the internet connection of the mobile devices and the adaptation system.

\textsuperscript{1}http://technical.openmobilealliance.org/tech/affiliates/wap/wap-277-xhtmlmp-20011029-a.pdf
From the conducted experiments, the authors concluded that the comic frame extraction method has 100% accuracy for flat comics\(^2\) and 91.48% for non-flat comics\(^3\). The accuracy for balloon detection and text extraction are 90.7% and 93.63%, respectively. There is a 10% accuracy improvement and 90% processing time improvement, compared to other methods.

CORVIS or ‘COMics Rendering system on Video Streaming’ is software, proposed in [19], which enables a user to generate comics based on a video stream, in a semi-automated manner. It takes a sequence of frames from a video stream and turns them into panels (as found in comic books). It will also add text and sound effects based on the audio from the video stream. To determine the main object of a cut, the authors used a manual method, hence the semi-automated manner mentioned earlier. With CORVIS, it is also possible to add speedlines for linear movement and rotation. To incorporate certain emotions (anger, joy, etc.) and scenery information (e.g., quietness) of the video in the comic, the software uses the typical background textures of comic books.

Continuing on the work conducted in [19], [30] suggests to use the information in the screenplay (dialogue, sound effects, etc.) to generate the content and shape for the text balloons and the sound effects for the panels. The proposed approach creates an automated solution, however, this solution still needs manual fine-tuning in the end.

### 2.2 Visualization

Enhancing the browsing of comics by using special thumbnails or visualizing a movie using a comic, these two examples already show that visualization regarding comics can have different meanings, and leads to different contributions of comics.

As more and more comics are being digitized: previously print-only comics are finding their way to the digital world and comics published for the first time are being released in both a digital and print version. However, this trend has not sparked the development of a search engine enhanced for the discovery of comics. The current systems only enable searching on basic metadata information (e.g., title and author). [17] suggests to create thumbnails for each comic to more easily identify the content of a single episode of a comic book series. The thumbnail is created by the following process on the input episode: frame segmentation of each page, followed by extraction of the features (e.g., edges to localize text) from every frame, used to calculate the importance score of the frame. The frames with the highest score are selected as part of the thumbnail of the episode. The authors compared their method against the conventional method for previewing an

\(^2\) Flat comics are comics where the characters and the text balloons stay in the boundaries of one panel.

\(^3\) Non-flat comics are the opposite of flat comics.
episode, namely viewing the first few pages of a comic. The results state that the proposed method enables users to search comic episodes faster, which is the intended goal of a search engine.

With *DigestManga*, the authors of [33] tried to develop a system to establish a smooth **relationship between movies and comics**. The more formal goals of DigestManga are: a visual summary of a movie and a summarized movie with comic editing, so users can visualize an input movie as a comic book and edit the movie through the visualized comic. It is possible to create visual summaries, based on comics, in a manual or automated way. If the user edits the comic, he/she can view the edited movie to experience the result. The system emphasizes the frames in the comic as features in the movie and the frames that do not appear as non-features will be played at a faster pace. DigestManga was well accepted by users because of the two kinds of summaries: the comic summary and the summarized movie. It was designed for beginners and not professional users, nevertheless it lacks several functions to make the output more interesting and comic-resembling. The advantage, as believed by the authors, is that the approach of DigestManga could be useful for services such as YouTube\(^4\) because their users can watch movies more interestingly and quickly, and beginners can start creating their own content (to upload it to the service).

The main problem [39] handles is how to display **high-resolution comic pages on low-resolution cellular phones**. Other problems that arise with the use of cellular phones are the use of a narrow bandwidth, together with the expensive cost, and necessity for low-size images because of the limited memory of cellular phones. The authors developed a system to circumvent these problems, that works as follows: frame detection is followed by layout analysis to determine the reading sequence. Next, text extraction is performed together with a layout analysis. The remaining drawn image is stored in a way enhanced for displaying on a cellular phone. For more details, we refer to [39]. Future work for the proposed system involves solving the sometimes poor quality of the images (due to the large pixel size of the frame), solving the insufficiency of the four level gray-scale images used to display tone patterns and improving poor image quality by using reproduction techniques of screen tone patterns.

A work dedicated to the **extraction of text** is [13]. It proposes a new online method for automatically extracting text inside text balloons of digital comic books. This method follows the following steps: frames are extracted from the input comic images, then the text balloons are extracted from the frames. With the balloons as input, the authors use text blob extraction (for more information, we refer to [13]) to extract the text. After conducting experiments, the authors concluded that the method has an accuracy of 100% for frame extraction for flat comics and 100% for balloon detection. **Text extrac-**

\(^4\)http://www.youtube.com
tion achieves 93.75% accuracy. The authors suggest that this method is useful for the automatic translation of Japanese comics into international comics.

Besides the digitization of comics, one could also enhance the reading experience of printed comics, as denoted in [31]. In this work, the authors use AR Comic Book (which runs on mobile phones) to add 3D augmented reality to comic books by using markers on the pages. These markers together with the application allow the reader to view virtual characters.

### 2.3 Metadata

Sophisticated functions for searching, accessing and browsing digital manga, together with the combination of these functions in a reading environment leads to very different functional requirements for metadata of digital comics compared to the traditional (printed) comics. [28] summarizes the basic requirements for a metadata framework:

- Different levels of description of the elements of a manga;
- A clear difference between the intellectual entity and the publication of a manga;
- Identification and description of elements that make up a manga, these intellectual entities form an ontological basis for the manga metadata.

These three requirements can be converted into three aspects: bibliographic, structural and ontology description. The authors suggest the use of Functional Requirements for Bibliographic Records for the bibliographic description, TV-Anytime for the structural description and Wikipedia’s description framework as a base for the ontology description. The authors were also working on the development of software tools for this metadata framework. Though, the development of the full metadata scheme was left as future work.

The Web is an important component in the creation and delivery of many manga and their related resources, however, this is not reflected in the creation of digital manga, because most of the time this is just a scan of the print manga, with minor editing for browsing on small(er) screens. During the production process of a manga, a lot of information is created. This information contains relationships among manga and their related recourses. During production, it would be useful if the management of these semantic connections is possible and this with an overhead that is as low as possible. Also, this linking should be independent of the output format of the manga, whether it is digital or print. [27] uses the Manga Metadata Framework (MMF) from [28]. The authors developed the Manga Path Language, which is based on the Resource Description Framework (RDF), to express the structural description of MMF. A metadata model to describe
the relationship between manga magazines and books, based on MMF’s bibliographic description, is also described in [27], together with a metadata model for the description of intellectual entities. Production support tools based on the MMF could also be developed, such as a digital manga storyboard editor, which enables the linking of resources, created in the production process, to the final manga product, and a manga collection viewer, which should be able to arrange the manga by different criteria (based on the models from MMF) and level of detail (based on the entities of the manga).

With the E-Comic Format (ECF), the authors of [25] developed a solution to overcome the management and consumption inconvenience of comic books. ECF uses the ISO base media file format\(^5\) to create the concepts of ‘image’ and ‘chapter’, it also enables multi-language support. MPEG-7\(^6\) metadata is used for the metadata information, under which they understand: bibliographic information, usage information and related content information. It also uses MPEG-21\(^7\) metadata for content protection and governance information. This allows for the control that only the users with a valid license are able to read the comic.

[14] presents TechCommix, which is an XML grammar and online comic computing environment. The authors wanted to incorporate comics as a way to translate traditional technical documentation into a more entertaining format. By using TechCommix, the difference between entertainment and instructional content can still be stated. TechCommix is based on ComicsML (more information about this format later) and Darwin Information Typing Architecture\(^8\) (DITA). In [14], also a description of a GUI to translate DITA tasks into comics, together with the XML file defined by TechCommix, can be found.

2.4 Accessibility

Storing information is one part of the complete picture, presenting this information is another one.

At the writing of [29], the digital presence of comics was minimal, most of them were solely available in print. These print versions restrict the access for a number of people: the visually impaired, motor-impaired and mobile device users. The authors want to enable those people to control the navigation through (digital) comic books in an automatic way. Currently available readers lack an easy way to control navigation, and mobile readers need to zoom in and out themselves. To enhance the navigation through a comic, the

---

\(^5\)http://mpeg.chiariglione.org/standards/mpeg-4/iso-base-media-file-format
\(^6\)http://mpeg.chiariglione.org/standards/mpeg-7
\(^7\)http://mpeg.chiariglione.org/standards/mpeg-21
\(^8\)http://docs.oasis-open.org/dita/v1.2/os/spec/DITA1.2-spec.html
solution proposed in [29] performs three steps: sequential ordering of the page-wide files, segmentation of each page into panels (using Watershed, for more information, we refer to [29]) and sequential ordering of the panels in the right reading order. From the validation conducted, the conclusion could be made that the algorithm is not perfect and has a number of limitations (e.g., handling special layouts). Through six scenarios, the user experience of the proposed approach has been validated. This resulted in the following suggestions: improving the interface and the accessibility (additional features, e.g., timed transitions and text-to-speech for text balloons).

With eDBtheque, the authors of [16] wanted to create the first comic books database, containing a ground truth, consisting out of a hundred pages with properties such as ‘published between 1905 and 2012’ and ‘balloons and text shapes are quite heterogeneous’. To build the ground truth, a visual segmentation protocol was developed with guidelines regarding text lines, balloons and panels. For the semantic information of the ground truth, the panels, balloons, text lines and the pages were annotated. This information is stored in a single SVG\(^9\) file, which enables the user to display the visual segmentation and/or semantic annotation of an object over a comic book page. During the evaluation, errors were discovered. However, these errors could be corrected manually by a post-production step. Together with the database, the authors also developed a tool, based on the work of Wolf [38], to evaluate the recall and precision of algorithms regarding the ground truth. Future work involves enhancing the ground truth, with the contribution of domain experts, to add, e.g., the types of balloons and the affiliation between objects.

2.5 Formats

A number of formats for digital comics are available, with each focussing mainly on one aspect of digital comics. The most commonly used formats for distributing comics online is the Comic Book Archive (CBA), where each comic is an archive (RAR, ZIP, etc.) containing image files (mostly PNG and JPEG). The images are viewed in ascending order based on the filenames of the images. The extensions corresponding with CBA are .cbr, cbz, etc. (determined by the archive type). The only real restricting is the supported image types of the CBA viewer. Because there is no standard specification for CBA, the assumption can not be made that a CBA viewer supports all or a certain image type.

Because CBA does not provide any facilities for incorporating metadata, ComicBook-Info\(^{10}\) was created. By making use of the ZIP comment available with ZIP archives, metadata information is added. This allows the creators to update the metadata without the need to unzip (and afterwards zip) the comic. For the readers, this means that they

\(^9\)http://www.w3.org/Graphics/SVG/
\(^{10}\)https://code.google.com/p/comicbookinfo/
can access the metadata information also without unzipping the CBA, which is required for reading the comic.

Besides ComicBookInfo, also other formats exist that model metadata information, such as Comic Book Markup Language (CBML). As described in [36], it is designed with the following goals in mind: support the study and analysis of comic books. To enable this, a database with digital comic books should be build, enhanced with encoded transcriptions and descriptive metadata. The latter is supported by CBML, together by the possibility to efficiently search a large database. The second motivation for creating CBML is the desire to explore more generally the modeling and representation of the broader class of documents that tightly integrate pictorial images and text. When creating this language, TEI\textsuperscript{11} (Text Encoding Initiative) was taken as a base because it provided options to support extensive analysis, textual criticism, and annotation. Also, the many available metadata structures lend themselves to describing the complex bibliographic histories and relationships that appear in comic books series, and across multiple series. The issue of copyright might form a problem because to build a CBML file copyright protected information is used. The same goes for building a database filled with comic books. The author suggests using comics that are not copyright protected or comics that entered the public domain. CBML can be enhanced, in future work, with additional visual and pictorial features. Also, a new version of CBML will include analysis frameworks and taxonomies. Besides the further development of the language, the author plans on building a database, using public domain comic books encoded in CMBL.

Opposed to adding metadata information to a format such as CBA, the goal of the next format is to create a ‘simple markup language for online comics’. ComicsML\textsuperscript{12} (Comics Markup Language) is a XML-based markup language for comics, developed by Jason McIntosh. The main reasons for this language are syndication, searchable archives, automatic index generation and equal access. As it is a markup language, the layout of a comic has not been incorporated in ComicsML. It gives information about the comic, however, it does not provide information about how to present the comic to the viewer.

Another format based on XML is the Advanced Comic Book Format\textsuperscript{13} (ACBF), designed to cover both the presentation and the metadata information of comics. The most important features of ACBF are: metadata information, comic’s structure, layering, semantics, multi-language support. A design choice worth mentioning is that the graphical and textual information is saved separately. The goals, to be achieved by ACBF, are: comic book’s content representation, automatic processing, indexing, collection management and formats conversion. One final note is that this format is DRM (Digital Rights Management) free and the specifications are open.

\textsuperscript{11}http://www.tei-c.org/index.xml
\textsuperscript{12}http://www.jmac.org/projects/comics_ml/about.html
\textsuperscript{13}http://acbf.wikia.com/wiki/Advanced_Comic_Book_Format_Wiki
Aside from the formats developed for comics, other formats are used to store and view digital comics. The *Portable Document Format*, or more commonly known as PDF, is used in a similar way as CBA: each page of a comic book is stored as one image/page in a PDF document. The main advantage of this approach is the support by almost every device now being used, ranging from desktop computers, to tablets, to smartphones. All of them have a PDF reader capable of reading these comics. The main disadvantages are the limited metadata information, limited accessibility, limited adaption to screen size, fonts etc., limited possibilities for animations and interactivity, and a limited possibility to monitor its use [11]. It is possible to store metadata information stored in PDF documents, however, this is hardly used and is limited to only the information of a standard document (e.g., title, author). This does not meet the needs of the metadata information of comic books.

Another format designed for digital publications, currently mostly books, is *Electronic Publishing* (EPUB), at the moment at version 3.0, which introduced, i.e., HTML5 and MathML support. There have been attempts to use EPUB to digitalize comics [4]. One sample is a Japanese manga where each page is represented by an HTML document having a single image. Another one uses the same manga, however, each page is represented by a single image, with a fallback HTML. This shows that it is possible to read comics in EPUB format, however, it lacks features (e.g., Guided View Technology [10]) that can be found in the dedicated apps by, i.e., comiXology, Marvel Comics or DC Comics.

Also, *proprietary formats* are being used to present digital comics. For example *iBooks Author format* (IBA), designed by Apple. This format is based on EPUB, however, it incorporates extensions (e.g., XML namespaces and extensions to CSS) to enhance the functionality. Of course, this makes it incompatible with readers who are able to handle the EPUB format. Comic writers already tried to use iBooks Author to create comics, resulting in approaches such as the ones seen with the EPUB format. In [24] another example of a comic in IBA can be found, where the author bundled two issues in a book and every issue is represented by a gallery holding each page as an image.

### 2.6 Conclusion

The work conducted in the different areas around (digital) comics is mainly based around the extraction of information. The need to resolve to extraction methods to get the
required information drives the research towards better and more efficient approaches and algorithms. These methods are necessary because the format in which the comic books are digitally stored resembles a physical comic. The first consequence is that almost no or limited metadata information is present, which hinders applications that require such information. Another consequence of the physical resemblance, is the limited possibility to manipulate the graphical information such as the panels and their characters. We discussed several formats to offer digital comics to readers. We conclude that most formats lack the necessary functionality for the presentation and/or the metadata of comics.
Chapter 3

Proposed Solution

In this chapter, we begin with listing the different requirements, in Section 3.1, needed to solve the problems of Section 1.1, that are being tackled by our solution. This is followed by the design choices made in this thesis, in Section 3.2. In Section 3.3, we give an overview of the proposed solution.

3.1 Requirements

The proposed solution exists out of two major elements: the presentation part and metadata part, which are discussed in Chapters 4 and 5, respectively. The requirements of the presentation part are

Req. 1 Animations,

Req. 2 Support different types of devices, together with

Req. 3 Reading assistance,

Req. 4 Audio, and

Req. 5 Multiple languages support.

The requirements of the metadata part are

Req. 1 The extension of administrative metadata, and

Req. 2 Support for descriptive metadata.
3.2 Design Choices

3.2.1 Rationale for EPUB Usage

As a foundation for the solution presented in this thesis, we choose the EPUB 3 format. The reasons for this decision are explained in the following sections.

Adaption

The EPUB 3 format is a widely used open format for distributing digital books [15]. Here, with ‘books’, we do not include comics. When shopping for books online a user sometimes has the option between different version of the same book, e.g., paperback, hardcover and digital. For example, Amazon offers different versions for a number of its books. A number of (online) stores offer just the EPUB version and solely depend on the sale of digital books, e.g., Apple’s iBookstore.

Purpose

The initial purpose of EPUB 3 is to support digital publishing, and not only the widely used, specific case of digital textbooks. Hence, the EPUB format should also be a vehicle for delivering digital comics to its readers. Almost no comics are available in the EPUB format, this, however, does not mean that it is not possible to offer comics in this format. This (specific) use has only been explored with limited efforts, which have not led to a useful approach for the digitization of comic books. A first effort represents each page by an HTML document having a single image. A second effort represents each page by a single image, with a fallback HTML. Both were mentioned in Section 2.5 and tested by us using the applications iBooks (version 3.2) and Calibre\(^1\) (version 1.9.0). The results of these tests were rather poor. We encountered problems displaying the version with HTML fallback in iBooks concerning the rendering of the images, however, with the other version we did not have this issue. With this last version Calibre had no issues, however, when displaying the HTML fallback version we were presented with a long text document, suggesting we were viewing the binary version of each image.

Technologies

The power of EPUB itself lies in the fact that it only uses Open Web Formats (OWF) such as HTML5, CSS3 and JavaScript. It has proven more that once that a lot of functionality is present in the different formats itself. Additionaly, there are an infinite

\(^1\)http://calibre-ebook.com
amount of possibilities when these formats are being combined. The Web is the best example. Aside from the presentation oriented formats such as the ones just mentioned, the draft of the new EPUB 3 specification includes formats to incorporate additional metadata information in an EPUB file [23]. These new formats are RDFa\(^2\) and Microdata\(^3\). More information about these formats and metadata itself can be found in Chapter 5.

With all the possibilities the format offers, we can conclude that at least 99% of the specifications, that make up the ‘perfect’ digital comic, can be accomplished by using one or a combinations of these OWF.

### 3.2.2 Localization of Metadata

The information stored in our solution will be limited to the information necessary for the processing of a single comic such as the list of characters appearing in a comic, in which panels of a comic a character appears, etc. and the basic linking of multiple comics. The more elaborate information is stored on a remote server, which can be requested if needed. This information will include metadata that is updated frequently. This removes the burden of updating the downloaded EPUB files on the users their devices.

### 3.2.3 Embedded and Inline Metadata

For the metadata stored locally, see the previous section, we choose to embed this information in the EPUB file itself. We added an additional file to contain the metadata that does not concern the panels and pages. The metadata about the panels and pages is provided inline in the files representing these elements.

### 3.3 Solution Overview

In Figure 3.1, we present a graphical overview of the solution, and how it is incorporated in a workflow. At the top left we start by the creation of the content, which is aided by an authoring environment that uses our solution. The authoring environment takes care of the digitization of the content, i.e., enhancing the presentation, providing functionality to add metadata information, and delivering the comic book in the format proposed in this thesis. On the right of the authoring environment you see a visual representation of our solution, consisting out of a presentation part and metadata part. The presentation requirements are listed on the left and the information provided by the metadata is listed on the right, together with the most importantly used technologies. At the complete right

\(^2\)http://www.w3.org/TR/rdfa-syntax/
\(^3\)http://www.w3.org/TR/2013/NOTE-microdata-20131029/
of the picture we see that the metadata is connect with *Dicera*, which is an ontology we created to structure the information regarding comic books, and their presentation. We connect a remote server with the metadata to enable the user to request more information if needed. At the top right of the figure we see that this format can be used by different platforms and applications, however, some might need to adapt their platform and/or application to support our solution. Nonetheless, this will only benefit their business.
Figure 3.1: Solution overview (©Apple, kobo & comiXology)
Chapter 4

Presentation

In this chapter, we talk about the first element of our solution: the presentation part. This includes everything that has to do with presenting the comic to the reader. In Section 4.1, we list the requirements, which we will tackle in detail in Sections 4.3, 4.4, 4.5 and 4.6. We introduce the concept of layering in Section 4.2. Next, we talk about the packaging of all the mentioned functionality in a reusable library, in Section 4.7. As will be explained in this chapter, we rely on JavaScript for a part of the functionality, however, what happens when scripting is disabled is discussed in Section 4.8. The unavailability of layers is a problem tackled in Section 4.9. During the development of our solution, new specifications for EPUB have been released, these are discussed in Sections 4.10 and 4.11. Finally, in Section 4.12, we end with a summary of this chapter.

In what follows we will use (basic) functionality of the OWF, which will not be discussed in detail. For an introduction to these standards, we refer to [5], [2] and [7]. Note that the examples given in this chapter are not complete: they are part of a complete example that will be discussed in Chapter 6.

4.1 Requirements

The first requirement is the support for all sorts of devices, with special attention to the mobile devices such as smartphones and tablets. The different screen sizes need to be taken into account when displaying a comic. Showing a complete page of a comic book on a smartphone does not allow the user to read the text inside the speech balloons and captions. The application should be able to zoom in automatically on every panel and follow the panels in the correct order, without too much intervention by the user, cfr. Guided View™ Technology. The next requirement is the possibility to add animations to comics. Instead of displaying a whole panel immediately, first the background could be shown, then the characters, followed by the speech balloons and their text. Including
minimal animations regarding text has been done by Marvel Comics in their ‘Avengers vs.
X-Men’ series issue ‘Avengers vs. X-Men #1: Infinite’. Accompanied with animations, a
creator of digital comics should also be able to add audio to a comic, whether it is music
or sound effects. The last requirement deals with the support of multiple languages.
If a publisher wants to sell comics in multiple languages, it would be beneficial that he/she
did not have to offer a separate comic for every language. Instead a single comic with
all languages in it should be offered to the reader, allowing him/her to switch between
different languages.

To summarize the aforementioned requirements, see the following list.

Req. 1 Animations

Req. 2 Support different types of device, together with

Req. 3 Reading assistance

Req. 4 Audio

Req. 5 Multiple languages support

4.2 Layering

Before we tackle the requirements, we define the way the graphical information is stored.
We do not store a page as one image anymore, however, we store each panel separately.
We take it even further: by using layering, we will store every panel, not as a single image,
however, as a group of images. Every image is called a layer, hence the name layering.
We can create a layer for the background, a layer for each character and a layer for each
text element (e.g., speech balloon, caption and effect). This allows us to do manipulations
on separate parts of the panel itself and not only on the whole panel. In what follows we
will explain how we use the OWF to support layering. Listing 4.1 gives a basic example,
using only HTML5.
This is fairly simple. We made the assumption here that all the layers are of the same size. If this were not the case, we had to manipulate their positions, however, this can be easily done using CSS3. Before using `<img>` elements to support layering, we first tried using the `<canvas>` element of HTML5. However, this approach was abandoned because it was too cumbersome (more information can be found in Appendix B).

### 4.3 Animations

To support animations in our solution, we rely again on the OWF. Building on the example of the previous section, we will show how it can be used to gradually build a panel by showing each layer one by one. The animation for showing a new layer is ‘appear’\(^1\). In Figure 4.1 we give an example animation sequence, using ‘appear’. Other animations are also possible. In Chapter 6 we will introduce the use of the ‘dissolve’ animation.

Starting from the example given in Listing 4.1, we first show the `<div>` element representing the background and hide all the other `<div>` elements. Every time an additional layer needs to be shown the corresponding `<div>` element is displayed. All this is done through the CSS property `display`, together with JavaScript.

\(^1\)The next layer appears on the panel, it does not move to its location, etc.
4.4 Reading Assistance

When talking about reading assistance we are talking about zooming in on the panel that the user is currently reading and navigating to the next panel once the whole panel has been read (determined by user input). To accomplish this, we make use of a jQuery\textsuperscript{2} plugin called Zoomooz\textsuperscript{3}. This JavaScript library enables to zoom in onto elements of Web pages. In our case, these are the panels of a comic. In Listing 4.2 we give an example with two panels. An important remark is that the reading assistance accomplished here is independent of the application, because it is achieved through scripting in the EPUB file itself, and can, hence, be different for every comic book.

Listing 4.2: Reading assistance example- panels

\begin{verbatim}
1 <div id='panel1'>
2     <div id='panel1_background' class='background'>
\end{verbatim}

\textsuperscript{2}http://jquery.com
\textsuperscript{3}http://jaukia.github.io/zoomooz/
<img src="panel1/background.png" alt="background image" onclick="showCharacterLayers('#panel1', event)"/>

<div id="panel1_character" class="character">
  <img src="panel1/character.png" alt="character cell image" onclick="showBalloonLayers('#panel1')"/>
</div>

<div id="panel1_caption" class="balloon">
  <img src="panel1/caption.png" alt="caption image" onclick="showTextLayers('#panel1')"/>
</div>

<div id="panel1_text" class="text_container" onclick="showNextPanel('#panel2', event)"
  <div id="panel1_text_obj" class="text_obj">
    <span id="panel1_text_nl" property="says" xml:lang="nl">Hoe het begon ...</span>
    <span id="panel1_text_en" property="says" xml:lang="en">How it began ...</span>
  </div>
</div>

<div id="panel2" >
  <div id="panel2_background" class="background">
    <img src="panel2/background.png" alt="background image" onclick="showCharacterLayers('#panel2', event)"/>
  </div>
</div>

<div id="panel2_character" class="character">
  <img src="panel2/character.png" alt="character cell image" onclick="showBalloonLayers('#panel2')"/>
</div>
When the page is loaded, the user is automatically zoomed in onto the first panel (see Listing 4.3).

Listing 4.3: Reading assistance example - window.onload

```javascript
window.onload=function(){
  showNextPanel('#panel1');
  $(window).click(function(evt) {
    if (evt.target == '[object HTMLHtmlElement]') {
      zoom('body', event);
    }
  });
}
```

Every time the panel is clicked on, another layer is made visible (see Listing 4.4).

Listing 4.4: Reading assistance example - show different layers

```javascript
//current language ('nl' or 'en')
var current language = nl;

//show all the characters layers from a given panel
function showCharacterLayers(panel, event){
  $(panel + '.character').show();
}
function showBalloonLayers(panel){
    $(panel + ' .balloon').show();
}

function showTextLayers(panel){
    $(panel + ' .text').show();
    $(panel + ' .text_' + current_language).show();
}

If the last layer has been made visible, the user moves on to the next panel (see Listing 4.5).

Listing 4.5: Reading assistance example - show next panel

function showNextPanel(panel){
    $(panel + ' .background').show();
    var background = panel + 'background';
    zoom(background, event);
}

After reading the second panel, and after clicking on that panel, zooming out allows to view the whole page. Clicking outside a panel, during reading, results in the same behavior (see Listing 4.6).

Listing 4.6: Reading assistance example - complete page read

function completePageRead(event){
    zoom('body', event);
}

The use of Zoomooz is shown in the function zoom(location, event), see Listing 4.7.

Listing 4.7: Reading assistance example - use of Zoomooz

function zoom(location, event) {
    event.stopPropagation();
}
For more information about the two calls done inside that function, we refer to the documentation of Zoomooz.

Other libraries are available offering the same functionality, however, we did not conduct a comparative study between them. Nonetheless, we were able to fulfill the requirements and determining the best or creating the ideal library is the topic of future work.

4.5 Audio

To add audio to the digital comic, we use the `<audio>` element of HTML5, as can be seen in Listing 4.8, together with JavaScript to trigger the playing of the audio, at the right moment.

Listing 4.8: Audio example

```html
<div id='panel_audio'>
  <audio id='audio' src='audio.wav' preload='auto'>
    Playing audio is not supported by your client.</audio>
</div>
```

Using CSS we can hide player, because the reader does not have to concern him or herself with this.

4.6 Multiple Languages Support

To enable multiple languages in the same digital comic and to allow the user or application to switch between the languages, we store the different translations of a piece of text in different ‘sublayers’ (`<span>` elements instead of `<div>` elements are used). This allows to show the (text) sublayers of the currently selected language and hide the other languages. (Un)hiding the different sublayers is again accomplished by JavaScript (for more details we refer to the example discussed in Section 6.1). In Listing 4.9 an example can be found of the use of two languages, Dutch and English. We have one layer representing all the text, which means both languages. Inside this layer we have another `<div>` element representing the text object. Why this is necessary can not be explained using this example, because it has to do with the metadata information that needs to be added later. This `<div>` element has two `<span>` elements (the sublayers previously mentioned),
one for each language. We use the `xml:lang` attribute to denote the Dutch and English translation. If, for example, we want to read the comic in English, we set the CSS property `display` to `none` of all the `<span>` elements with the value of the `xml:lang` attribute set to `nl`. This makes those elements hidden, leaving only the English translation visible. Using CSS we can also set the font of the text, together with the size and the position on the panel.

Listing 4.9: Multiple languages support example

```html
1 <div id='panel_text'></div>
2   <div id='panel_text_obj'></div>
3       <span id='panel_text_n1' xml:lang='nl'>Hoe het begon
3           ...
4       </span>
5       <span id='panel_text_en' xml:lang='en'>How it began
5           ...
6     </span>
7 </div>
```

### 4.7 Use of Scripting Languages

As the attentive reader will have noticed, most of the JavaScript code is not inherit to a single comic, however, it is reusable for a lot of comics. In this thesis, we suggest that it is useful to create a JavaScript library to bundle all the reusable code. First, we list the requirements of such a library. Second, we present our own implementation, called `comicreader.js`.

The first part of the library should consist out of the functions to provide the animations. Besides the standard ‘appear’ animation, other (basic) animations could also be added, such as slide, dissolve, etc. The functions should also be designed in a way that they can be reused if a user (here, a developer and not the reader) wishes to create its own animations. The second part is concerned with the reading assistance, for which we now rely on the Zoomooz library (and jQuery). Another part of the library handles the support for different languages and more specific: the switching between the languages by manipulating the different layers.

Besides listing the different functionalities the library should offer, we already want to address one design issue: the automatic detection of the reading order of each panel. As will be discussed in Section 5.2.2, we will incorporate this information in the metadata, however, for the presentation of the comic we also need this information. There are two options. The metadata is used or another way of providing this information is created. Why would we add this information twice, and thereby create redundancy? The infor-
information provided by the metadata solution creates an indirection, hence we will need to ‘parse’ more code of the XHTML pages before we get the desired information, which will create a delay. The question (and what needs to be studied) is if this makes any difference: does the indirection create a (significant) delay or not and if it generates a delay, can this problem be bypassed by reading all the information concerning the reading order when the EPUB file is loaded. This will increase the delay while opening a file, however, no delay will be experienced when reading the comic book. If the load time or the delay creates a problem, we can add a \texttt{data-readingorder} attribute to the \texttt{<div>} tag of a panel, which allows us to circumvent the indirection and increase the performance.

The common functionality of the pages from the example EPUB file presented in Chapter 6 is grouped in a JavaScript library \texttt{comicreader.js}. It does not qualify to all the requirements stated above, however, it already shows the usefulness of developing a dedicated JavaScript library.

4.8 Fallback When Scripting Is Disabled

As can be found in the specification of EPUB 3, relying on scripting to deliver the content to its users is not allowed. It can only be used to enhance the experience. Hence, we need to have a fallback in case scripting is not possible or (temporarily) disabled on a device (or in an application). Our fallback method displays the whole page at once. We do not use layering anymore, there are no animations, no reading assistance, no audio and no multi-language support. We included for every panel one image as a fallback, which will be displayed when it is not possible to execute scripts. The text on each of those panels is hard coded, so a creator has to choose a language in advance, it can not be altered at runtime (see Section 4.11 for more information about this topic). In Listing 4.10 an example can be found of a normal panel together with the fallback panel. The fallback panel will be shown by default, however, when scripting is enabled, those fallback panels will be removed and replaced by the normal panels. This will be accomplished by CSS and the \texttt{display} property, as can be seen on lines 32 and 33 of the example. The function connected to \texttt{window.onload} will only be called when JavaScript is enabled.
Listing 4.10: Fallback example

```html
1 <div id='panel_fallback' class='panel_fallback'>
2     <img src='fallback.png' alt='panel fallback image'/>
3 </div>

4 <div id='panel' class='panel'>
5     <div id='panel_background' class='background'>
6         <img src='background.png'/>
7     </div>
8
9     <div id='panel_character' class='character'>
10        <img src='character.png'/>
11     </div>
12
13     <div id='panel_caption' class='balloon'>
14        <img src='caption.png'/>
15     </div>
16
17     <div id='panel_text' class='text'>
18         <div id='panel_text_nl' class='text_nl'>
19             Hoe het begon ...
20         </div>
21         <div id='panel_text_en' class='text_en'>
22             How it began ...
23         </div>
24     </div>
25 </div>

26 <script>
27     // For brevity, not all JavaScript is included here.
28     window.onload=function(){
29         showNextPanel('#panel');
30         $('.panel_fallback').hide();
31     }
32 </script>
```
4.9 Layer Unavailability

In the previous sections, we assumed that a comic (or more specific every panel) is available in layers. What happens when this is not the case or if an older comic (without layering) needs to be converted into the form specified in this thesis? In this case, we assume that every panel is represented by a single image.

Adding animations to a single panel will not be possible, because the different layers to work with are not available. Multiple languages support is not possible because the text is hard coded on the image. Audio can still be added to the comic, together with reading assistance.

What the influence is of the unavailability of layers on the metadata is a part of the discussion in Chapter 5.

4.10 EPUB Region-based Navigation Specification

The EPUB 3 (draft) specification about region-based navigation [22] talks about enhancing the visual rendition of EPUB publications using regions of interests. This was discussed during the International Digital Publishing Forum Workshop on Sequential Art in Paris on March 26, 2014. The specification’s goal is to allow reading progression at the sub-page level, which is considered an important requirement for comic books and magazines. They want to allow the user to navigate between the panels of a comic in predefined order, together with an enhanced presentation of those panels. Instead of showing the complete page, the reading system zooms in on the current panel. This is analog to the topic of Section 4.4. Aside from zooming in on a panel, the specification also discusses zooming in on different regions of the panels, e.g., speech balloons. Zooming in on the speech balloons or other regions of interest inside a panel is not a topic of this thesis, however, it is possible using a similar approach to the one discussed in Section 4.4.

To accomplish the goals tackled in the specification, they introduce the guided nav element. Using this new element, allows to determine the different regions of interest of a comic book page, which is represented using a single image. This is immediately the biggest different between our approach and the approach defined in the specification. We split a page into panels and each panel consists out of different layers. The approach of the specification can deal with the fact if a page is split in different images for each panel, however, layering is not part of the specification, and is in our solution necessary to allow animations (see Section 4.3).

http://idpf.org/idpf-comics-manga-workshop-paris
This specification might offer a solution to one of the problems mentioned (and not solved in this thesis) in Section 4.7. The major problem with this specification is the lack of layering, hence the inability to deal with animations.

4.11 EPUB Multiple-Rendition Publications

Another topic discussed during the workshop, mentioned in the previous section, is multiple-rendition publications [21]. This (draft) specification discusses the adaptation of an EPUB publication to deal with the different settings of reading systems. This does not only include the visualization of the content, however, it also includes the adaptation of the actual content that is being displayed. One example is the use of multiple languages. Instead of creating a different EPUB publication for every language, a different rendition for each language is included in the same EPUB publication. The specification talks about the processing model that a reading system follows to determine the best rendition. Another topic is the mapping between renditions, because one wants to allow the user to move from a certain spot in one rendition to the same spot in the other rendition. An example of this can be switching between portrait and landscape mode, where the user wants to continue reading at the same spot after turning his/her device.

How does this affect the solution proposed in this thesis? The first example given above is the use of multiple languages and is also a topic of this thesis (see Section 4.6). The difference between our approach and the approach proposed in the specification is that every language should point to a different rendition. Our solution uses one file for each page that includes all the languages. Splitting those languages in different files would force a developer to duplicate all the other code inside the files except for the actual text of the comic. Hence, redundancy is created and the file size of the EPUB publication will increase (unnecessarily). Studying different approaches is proposed such as using renditions with multiple languages using one file including all the languages, together with the use of JavaScript. Each rendition uses a different script that displays the correct languages and hides the other languages (analog to the approach in Section 4.6). The downside is that this approach brings us to the problems that might arise if scripting is disabled (see Section 4.8). This is not an issue when using the approach described in the specification.

In this thesis we talk about reading assistance (see Section 4.4). This can also be enhanced using multiple-rendition publications. For example, reading systems can make the distinction between portrait and landscape mode, followed by loading the correct rendition. This approach removes the burden, from the JavaScript library, of determining the orientation of the device.
We can conclude that a number of the goals of this specification are addressed in this thesis and that the solution proposed in this thesis can be complemented by this specification.

4.12 Summary

This chapter discussed the presentation part of the solution proposed in this thesis. We included features into our solution such as layering, animations, reading assistance, audio and multiple languages support, and this while resolving only to the functionality offered by the EPUB 3 format and the OWF. We discussed the use of (static) fallback panels when scripting is not enabled, and the consequences of not having scripting enabled and the unavailability of layers. The current work being conducted, by the group concerned with the development of the EPUB 3 standard, spends more attention to different types of publications, besides textbooks, such as comics. The region-based navigation and multiple-rendition publications are two good examples. Work regarding the integration of these specifications in our solution should be the topic of future work.
Chapter 5

Metadata

After addressing the first part of our solution: the presentation part, we now discuss the second big part: the addition of machine-understandable metadata. In this chapter we will first talk about the need for metadata in Section 5.1 and explain the different types of metadata, together with how they will be incorporated in the EPUB format in Section 5.2. In Section 5.3, we discuss how we connect these two types of information. Analogous to Chapter 4, we will also discuss the issue of layer unavailability in Section 5.4. In Section 5.5, we give a summary of the chapter.

The examples given in this chapter are not complete: they are part of a complete example that will be discussed in Chapter 6.

5.1 The Need for Metadata Information

The EPUB format allows its authors to include metadata information. This is mainly administrative metadata e.g., title, author, publisher, etc. The package.opf file, included in every EPUB file, contains this information. For more information about all the metadata that can be included in this file, we refer to its specification [20]. Aside from the information made available through the package.opf, we also want to add descriptive metadata information, e.g., information about characters, locations, story arcs, genres, etc., together with the necessary administrative metadata to manage these concepts.

A concrete example is the representation of a comic universe. One can consider the Marvel Universe\(^1\) as an example. This universe has over 5000 characters\(^2\), together with a large number of story arcs, that might come together at a certain point and split again (later). The discovery of new comics and story lines would be easier if one had the possibility to visualize the universe. Though useful for new readers, this would also

\(^1\)http://marvel.com/universe/Main_Page
\(^2\)http://marvel.com/characters
enable the fanatics to have a visual (and understandable) overview of the universe they love. Besides the interaction of the different story arcs, this approach could also be used to incorporate movies\(^3\) and video games\(^4\) starring the Marvel heroes such as Spiderman and the Avengers. This would enable readers to view the similarities between the comics and the cinematic interpretation of the story, together with which part of the story made it into the movie. All this information allows the user to synchronize his/her reading progress with the movie timeline, enabling the user to switch between the movie and the comic.

The above example allows us to summarize the advantages of metadata, towards the content, in the following list.

- Searchability
- Discoverability
- Maintainability
- Connectability

5.2 Different Types of Information

We can divide all metadata information into two groups based on the location where the information is stored: locally or remotely. With remote information we denote all the information that is not necessarily stored inside the EPUB file itself. A publisher can store all the information of a certain comic character on a server, e.g., biography, all the comics he/she appears in, etc. Ideally, the information stored in an EPUB file is not that extensive, e.g., the name of the character. This approach could be compared to the use of RSS readers. RSS reader retrieve information about the latest posts of a website. This information consists normally out of the title, the date and a short summary. For more information about the post, the RSS reader (upon action of the user) needs to request more information from the website or redirects the user (using a browser) to the website with the post. The most important information is stored locally and additional information is requested, from a remote location, if necessary.

5.2.1 Remote Information

Why store information remotely and not everything locally in the EPUB file? First, storing all the information would add to the size of the EPUB file. Second, this creates

\(^3\)http://marvel.com/movies/all
\(^4\)http://marvel.com/games/all
redundant information, i.e., what to do when there is an error in the biography, how is the EPUB file updated on the device of the user? When new comics are being published how is the information in the EPUB files, that are already downloaded, updated?

5.2.2 Local Information

The information stored locally is limited to the necessary information to enable the user to gain at least the minimum advantages over having no metadata information. In what follows we will first list all the information that is present and second, we’ll give more information about each element in the list. The list is as follows.

- Character
- Location
- (Special) object
- Story arc
- Issue
- Genre
- Content rating

With a location we denote the location of a scene in a panel. A (special) object concerns objects that have a special story or that need extra information, i.e., if a treasure (the object), that appears in multiple issues, needs to be retrieved it should be represented using this concept. A story arc, as can be found on Wikipedia\(^5\), is an extended or continuing storyline in episodic storytelling media such as television, comic books, comic strips, boardgames, video games, and in some cases, films.

For every character the name (in multiple languages) can be added, we can denote whether he/she appears on the cover and denote the global identifier. For a location and a (special) object, the same information is available. Every comic can be associated with a certain story arc. This happens through an issue entity, which also includes the issue number. For the genres (which can be more than one for a single comic) the name (in different languages) can be given and how many percent it matches, because they can be quantified, i.e., a story can be only 25% drama (not necessarily 100%). Rating the content is done by specifying the system that is used for rating (e.g., ‘Marvel Content Rating’) and the actual rating (e.g., ‘T’, for Teen). We will illustrate these concepts using an example in the next section.

\(^5\)http://en.wikipedia.org/wiki/Story_arc
To structure the concepts explained in the previous section, we have built an ontology using RDF. The ontology is called **Dicera** (the DIgital Comic book ERA vocabulary). We choose RDF because it can be used together with RDFa (more specific RDFa Core 1.1 [35]), which enables us to add metadata to Web documents. As mentioned in Section 3.2.1, it is allowed in the new EPUB 3 (draft) specification to add RDFa tags to the XHTML pages.

RDFa is not the only option allowed by the specification to add metadata: we could also use Microdata. The reasons why we choose for RDFa over Microdata are listed next. First, Microdata does not allow to easily use more than one vocabulary at the same time. To accomplish such a thing, residing to less convenient (hack-like) methods is the only possibility, i.e., adding another metadata language (such as RDFa), using full URL’s for the not-default vocabularies or even duplicating data! RDFa allows multiple vocabularies through the prefix attribute. Second, RDFa is supported by all major search engines such as Google, Bing, Yahoo! and Facebook’s Open Graph Protocol (OGP). Microdata is supported by all of these engines except Facebook. Hence, the use of Microdata and requiring support by Facebook forces also the use of RDFa. Only using RDFa provides support by all four of those engines and only one format needs to be used. The last advantage of the use of RDFa is that it allows our solution to work together with Linked Data Fragments servers and clients [34].

For the full ontology in RDF format, we refer to Appendix A.1 or [http://semweb.mmlab.be/ns/dicera](http://semweb.mmlab.be/ns/dicera). Although the EPUB format already has the package.opf file to present (some) metadata information using the XML format, we are not able to add our own concepts, hence we need to resolve to other resources such as the XHTML pages and RDFa. We also add an additional XHTML page, called metadata.xhtml, which contains the information about the characters, locations, objects, genres, issues, content ratings and the story arc. It does not include information about in which panel each of those characters, locations and objects appear. That information will be included in the other XHTML pages (which contain the panels).

In our first example we will show how we define the elements just mentioned. In the opening tag of the `<body>` element we define the vocabulary we use (see Listing 5.1), which is the ontology mentioned earlier.

[Listing 5.1: Metadata.xhtml example - vocab](#)

---

6[^6]: [http://www.w3.org/wiki/Mixing_HTML_Data_Formats#Mixing_Vocabularies_in_microdata](http://www.w3.org/wiki/Mixing_HTML_Data_Formats#Mixing_Vocabularies_in_microdata)
7[^7]: [https://support.google.com/webmasters/answer/146898?hl=en](https://support.google.com/webmasters/answer/146898?hl=en)
10[^10]: [http://ogp.me](http://ogp.me)
In Listing 5.2 the definition of a character called ‘Tante Sidonia’, from the famous Belgian comics called ‘Suske en Wiske’, can be found. Using the resource tag, we define the name by which we will identify this character during the whole story. We also need to declare the type (using typeof) of this resource, which in this case is a character. On line 2 and 3, the English and Dutch name of this character are presented. The global identifier is given on line 4.

Listing 5.2: Metadata.xhtml example - Character

```html
<div resource='#character_sidonia' typeof='Character'>
  <span property='hasName' xml:lang='en'>Aunt Sidonia</span>
  <span property='hasName' xml:lang='nl'>Tante Sidonia</span>
  <span property='hasGlobalIdentifier'>com.company.character.sidonia</span>
</div>
```

The same can be done for the location denoting Belgium and the story arc (see Listing 5.3).

Listing 5.3: Metadata.xhtml example - Location & Storyarc

```html
<div resource='#location_belgium' typeof='Location'>
  <span property='hasName' xml:lang='en'>Belgium</span>
  <span property='hasName' xml:lang='nl'>Belgi</span>
  <span property='hasGlobalIdentifier'>com.company.location.belgium</span>
</div>
```

```html
<div resource='#storyarc' typeof='Storyarc'>
  <span property='hasName' xml:lang='en'>Suske and Wiske</span>
</div>
```
In Listing 5.4 we define the two genres of the comic. The special property of these resources are the `matches` property (lines 4 and 9), as explained earlier.

Listing 5.4: Metadata.xhtml example - Genre

```xml
<div resource="#genre_fiction" typeof="Genre">
  <span property="hasName" xml:lang="en">Fiction</span>
  <span property="hasName" xml:lang="nl">Fictie</span>
  <span property="matches">75</span>
</div>

<div resource="#genre_drama" typeof="Genre">
  <span property="hasName" xml:lang="en">Drama</span>
  <span property="hasName" xml:lang="nl">Drama</span>
  <span property="matches">25</span>
</div>
```

The rating of the comic is defined in Listing 5.5, where the system that is being used is denoted on line 2 and the actual rating can be found on line 3.

Listing 5.5: Metadata.xhtml example - ContentRating

```xml
<div resource="#content_rating" typeof="ContentRating">
  <span property="usesSystem">Marvel Content Rating</span>
  <span property="hasRating">T</span>
</div>
```

In Listing 5.6 we define a special object representing a treasure.

Listing 5.6: Metadata.xhtml example - Object

```xml
<div resource="#object_treasure" typeof="Object">
  <span property="hasName" xml:lang="nl">Schat</span>
  <span property="hasName" xml:lang="en">Treasure</span>
  <span property="hasGlobalIdentifier">com.company.object.treasure</span>
</div>
```

We also define the issue of the comic, and connect the issue to a story arc and state the issue number (see Listing 5.7).
Now we have defined all the elements of the comic, we need to connect them to a comic object. This is done using the properties isIssue, talksAboutCharacter, talksAboutLocation, talksAboutObject, hasGenre, ContentRating and hasPage (see Listing 5.8). The property hasPage is used to connect the concept of a comic to its pages that present the content. More information about pages is given in the next example.

In our second example we will add more information to a page and a panel. In Listing
5.9 we define the vocabularies we are using. The one defined by the **vocab** attribute
defines our main vocabulary, Dicera. The ontology defined by **prefix** connects **emotion**
to an ontology called ‘Emotion Ontology’\(^1\) developed by the Swiss Centre for Affective
Sciences, in collaboration with the University at Buffalo.

Listing 5.9: Panel metadata example - **vocab** & **prefix**

```html
<body vocab="http://semweb.mmlab.be/ns/dicera#" prefix="
emotion: http://purl.obolibrary.org/obo/>

In this example we only look at one page, see Listing 5.10, which is the first page
of our comic. This page has six panels. These panels are connected to this page using
**hasPanel** as can be seen on lines 4 till 9. On line 3 we see that this is the first page that
has to be read.

Listing 5.10: Panel metadata example - **Page**

```html
<div id="page1" typeof="Page" resource="#page1">
  <span property="readingOrder">1</span>
  <span property="hasPanel" resource="#panel1"></span>
  <span property="hasPanel" resource="#panel2"></span>
  <span property="hasPanel" resource="#panel3"></span>
  <span property="hasPanel" resource="#panel4"></span>
  <span property="hasPanel" resource="#panel5"></span>
  <span property="hasPanel" resource="#panel6"></span>
</div>

We only show and discuss the first panel, because the other panels are similar. The
first panel that has to be read is ‘panel1’, as defined on line 3 of Listing 5.11. It has 7
layers, as defined on lines 4 till 10.

Listing 5.11: Panel metadata example - **Panel**

```html
<div id="panel1" typeof="Panel" resource="#panel1">
  <span property="readingOrder">1</span>
  <span property="hasLayer" resource="#layer1"></span>
  <span property="hasLayer" resource="#layer2"></span>
  <span property="hasLayer" resource="#layer3"></span>
  <span property="hasLayer" resource="#layer4"></span>
  <span property="hasLayer" resource="#layer5"></span>
  <span property="hasLayer" resource="#layer6"></span>
</div>

\(^1\)https://code.google.com/p/emotion-ontology/
In Listing 5.12 we define two balloons and to which character they belong.

Listing 5.12: Panel metatdata example - Balloon

The first layer we encounter contains the background image of our panel, together with the location information about the panel (see Listing 5.13).

Listing 5.13: Panel metatdata example - Layer with background

The next two layers, as can be seen in Listing 5.14, contain two characters. These are followed by two layers containing their speech balloons.

Listing 5.14: Panel metatdata example - Layer with characters and balloons
The last two layers are the ones containing the text said by the characters (see Listing 5.15). The two properties here worth mentioning are isSaidThrough and withEmotion. The first one connects the text to a speech balloon (which can than be used to determine the character who says the text) or a caption. The second one denotes the emotion expressed by the text. Both translations of the text are presented by a <span> element and the property attribute with value says. The language of each translation is denoted by using the xml:lang attribute.

Listing 5.15: Panel metatdata example - Layer with text

```xml
<div id="panel1_text_a" typeof="Layer" resource="#layer6">
  <span property="hasText" resource="#panel1_text_a_obj"></span>
</div>
```

11 12 13 14 15 16 17 18 19
Through the RDF schema we defined the following property `isBalloonFrom` to denote to which character a certain balloon belongs. For our use cases (see Chapter 6) this works, because we search first for the balloons (we search first for the text to be cor-
rect) and then ask to which characters they belong. What happens if we first select a character and then want to get all the balloons belonging to this character? Using only the isBalloonFrom property we would need to iterate all the balloons and check if the connected characters matches our character. This is not very intuitive. We would like to have a property hasBalloon, which denotes all the balloons belonging to a certain character. Introducing this property in the RDF schema would require us to define two properties that are actually each others inverse and this creates redundancy. OWL\(^{12}\), or Web Ontology Language, brings a solution to this problem, by allowing use to add more semantics to our schema. With the use of inverseOf, we can define the inverse of a property. In our case, we say that hasBalloon is the inverse of isBalloonFrom. If we only use isBalloonFrom in our XHTML pages, we can still use the property hasBalloon because we know now that it is the inverse of isBalloonFrom. For the full ontology in OWL format, where we introduce the use of inverseOf, we refer to Appendix A.2 or http://semweb.mmlab.be/ns/dicera.

**Schema.org**

Schema.org\(^{13}\) offers a collection of schemas to add metadata information to Web pages, with the goal of enhancing the recognition by search engines. When designing Dicera, we considered building upon those schemas. In Table 5.1 we compared the (most important) classes we defined in our ontology with the available classes at Schema.org. If it would be favorable to connect one of our and Schema.org’s classes it will be stated in that table, however, we did not incorporated this in our ontology. This is a topic for future work. From the comparison, we conclude that Schema.org can be used, however, a number of classes are lacking, which results in the absence of a number of necessary properties.

\(^{12}\)http://www.w3.org/TR/owl-ref/

\(^{13}\)http://schema.org
<table>
<thead>
<tr>
<th>Dicera</th>
<th>Schema.org</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Explanation</td>
</tr>
<tr>
<td>Character</td>
<td>Person</td>
</tr>
<tr>
<td></td>
<td>Too limited, a character can also be an animal or alien.</td>
</tr>
<tr>
<td>Location</td>
<td>Place</td>
</tr>
<tr>
<td></td>
<td>Should be considered; Only considers places on earth, globalLocationNumber might be not as good enough as our Global Identifier.</td>
</tr>
<tr>
<td>Object</td>
<td>Thing</td>
</tr>
<tr>
<td></td>
<td>The most generic item of Schema.org and an object in Dicera can be everything.</td>
</tr>
<tr>
<td>Content rating</td>
<td>contentRating property of CreativeWork</td>
</tr>
<tr>
<td></td>
<td>This property is only a String and Dicera uses an object to separate the system and the rating.</td>
</tr>
<tr>
<td>Genre</td>
<td>genre property of CreativeWork</td>
</tr>
<tr>
<td></td>
<td>This property is only a String and Dicera uses an object to separate the name and the match.</td>
</tr>
<tr>
<td>Issue</td>
<td>Not found</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Story arc</td>
<td>Not found</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Comic</td>
<td>Extension of Book or CreativeWork</td>
</tr>
<tr>
<td></td>
<td>Too limited, more comic specific information needed</td>
</tr>
<tr>
<td>Page</td>
<td>Not found</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Panel</td>
<td>Not found</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Layer</td>
<td>Not found</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 5.1: Dicera vs. Schema.org

**FOAF**

The *Friend of a Friend (FOAF)* project has the goal, as stated on their website\(^{14}\), to create a Web of machine-readable pages describing people, the links between them and the things they create and do; it is a contribution to the linked information system known as the Web. The same question as with Schema.org arises: can we build upon the FOAF vocabulary? From the comparison in Table 5.2, we conclude that this vocabulary lacks the necessary classes, and because almost no classes are represented, (almost) none of the required properties are available. This make us conclude that FOAF is **too limited to be used for our purposes**.

\(^{14}\)http://www.foaf-project.org
<table>
<thead>
<tr>
<th>Dicera</th>
<th>FOAF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Explanation</td>
</tr>
<tr>
<td>Character</td>
<td>Person</td>
</tr>
<tr>
<td>Location</td>
<td>Not found</td>
</tr>
<tr>
<td>Object</td>
<td>Not found</td>
</tr>
<tr>
<td>Content rating</td>
<td>Not found</td>
</tr>
<tr>
<td>Genre</td>
<td>Not found</td>
</tr>
<tr>
<td>Issue</td>
<td>Not found</td>
</tr>
<tr>
<td>Story arc</td>
<td>Not found</td>
</tr>
<tr>
<td>Comic</td>
<td>Document</td>
</tr>
<tr>
<td>Page</td>
<td>Not found</td>
</tr>
<tr>
<td>Panel</td>
<td>Not found</td>
</tr>
<tr>
<td>Layer</td>
<td>Not found</td>
</tr>
</tbody>
</table>

Table 5.2: Dicera vs. FOAF

5.3 Connecting Local and Remote Information

To connect the information stored locally and remotely, we use the global identifier, introduced in Section 5.2.2, which is the needed additional administrative metadata to manage the newly introduced concepts. This identifier, associated with, e.g., a character, can be used to retrieve information about a character that is not stored in an EPUB file, e.g., in which comics that character appears. An application that allows to read EPUB files can be extended to deliver more functionality not only through the information stored in the EPUB files itself, however, also by requesting additional information from the remote server, using the global identifier. This creates, for the user, the opportunity to discover new information about certain aspects of his/her comics. How the information of the remote server is known by the reading system is part of future work (see Section 9.2).

More information about concrete use cases, using metadata, can be found in Section 6.2.

5.4 Layer Unavailability

Layer unavailability puts limitations on the presentation part of our solution. The limitations on the metadata part are less severe. Each panel can still be annotated
with his character, however, all characters will be included on the same layer. The same goes for the location, text elements and special objects. Even if the text is hard coded on the image, text objects can still be used. These objects will solely be used for metadata purposes and not for both presentation and metadata as in the normal case. It is still possible to denote the location of a panel.

The only real limitation is the following. If the metadata is used to determine all the panels where a character is present, it is only possible to get the complete panel/image. When layers are present, it is possible to retrieve images containing only the requested character, without the background, the other characters, the text elements and the special objects.

5.5 Summary

In this chapter, we discussed the metadata part of our solution. We described the need for additional administrative metadata, and the introduction of descriptive metadata. This metadata information is divided in two groups based on the location where it is stored, i.e. locally or remotely. We introduced our ontology Dicera, used to structure a comic book’s information, in both RDF and OWL format. We elaborated on the way the local information is stored, using RDFa, and how it is connected to the remote information, using a global identifier. We concluded that the consequences of layer unavailability for the metadata part are less severe than those for the presentation part.
Chapter 6

Implementation

In this chapter we discuss multiple (complete) examples/use cases to illustrate the techniques and methods discussed in the previous chapters. In Section 6.1, we will give an implementation of a digital comic where we focus on the presentation part of the solution presented in this thesis. In Section 6.2, we focus on the use of the metadata, through two use cases and a desktop application. Section 6.3 provides a summary of the chapter.

On the CD accompanied with this thesis, all the files can be found to reconstruct and study the examples and use cases given in this chapter.

6.1 Presentation

To showcase the functionality discussed in Chapter 4, we created a digital comic from scratch. The comic contains three pages where we will emphasize different features on each page. All pages have reading assistance enabled and support multiple languages (however, the third page uses no text). The first and second page use the ‘appear’ animation, where the third page focuses more on animations including the dissolve animation and the movement of a character inside a panel. The first page has also audio embedded in it. The only difference between the first and the second page is the use of speech balloons and captions, respectively. The EPUB file of this example is called ‘comic.epub’. The files c1.xhtml, c2.xhtml and c3.xhtml contain the first, second and third page, respectively. These files can be found in the EPUB file, in the folder EPUB. The techniques and methods used in these files are the same as discussed in the previous chapter, so elaborating on these is not necessary.

Validation The generated EPUB file is valid conform the latest EPUB 3 specification, with exception of the tags used for RDFa, because their use is only a draft at the moment (see Section 3.2.1), however, this should not be a problem in the future.

51
Metadata  Aside from the presentation part in this EPUB file, the metadata is also present, using RDFa. Hence, this EPUB file can also be used to test the usage of the metadata information.

6.2 Metadata

In this section, we will present two different use cases and a desktop application, based on the metadata information found in the EPUB files, as presented in this thesis. The first use case (called ‘Character-Driven Comic Search Engine with Geographical Visualization’) will focus on the global information contained in every EPUB file: the characters and locations appearing in the comic. The second use case (called ‘Character-Driven Panel Extractor for Comics’) will focus on the information included in the XHTML pages containing the content of the comic: the characters appearing on each panel. The desktop application extracts detailed information from an EPUB file and allows the user to fetch additional (and more detailed) information.

For both use cases we used 10 comics of ‘Suske en Wiske’, converted into the EPUB format (by Joachim Van Herwegen, using a method based on Watershed [29]; The whitespace on a page are filled, and the planes that are not being filled are considered panels. The planes are checked whether they are convex or not. If this is not the case the planes are cut at the non-convex points, resulting in two new planes.). For each comic, we annotated the first 30 panels. We also added the metadata.xhtml file (see Section 5.2.2), together with the relevant information for each comic. To construct these two use cases, we created two parts. First, we created a Java application that will handle the reasoning, i.e., reading the EPUB files and extracting the necessary information. The second element is a website that presents the information to the user in a way that he/she does not know where the information comes from and how it is being processed. The Java application provides a REST API to allow the website to send queries to the application. For more information about this Java application, we refer to Appendix C.

For the desktop application we used the comic presented in Section 6.1, which has all the features our solution offers. To implement the application, we used Java.

6.2.1 Character-Driven Comic Search Engine with Geographical Visualization

This use case focuses mainly on the information provided by the metadata.xhtml file. It allows the user to search for comics based on (a part of) the name of a character, together with a minimum percentage of on how many panels that character actually
should appear (see Figure 6.1). If, for example, we have 30 panels in a certain comic and the selected character appears on 15 panels, then we will only select that comic if the minimum percentage was set to 50% or lower. Because the user might find it cumbersome and counter intuitive to work with percentage we present this feature using three labels called ‘Low’, ‘Medium’ and ‘High’, which map to the following ranges of percentages: [0-100], [35-100] and [67-100], respectively. Searching on the name of the character is not case sensitive. After the user hits the Search button all the necessary information is send to the Java application. The result is sent back and is processed by the website. The user can choose how the result is being displayed. He/she can view the results as a grid of comic book covers (see Figure 6.2). The other option is the use of a world map that will mark all the locations where the comics, containing the selected character, take place. Selecting a location on the map will display all the comics that take place there (see Figure 6.3). This showcases the power of the metadata regarding the location information. The website can be found on the CD in the folder code/website/characters.

6.2.2 Character-Driven Panel Extractor for Comics

This use case uses information embedded in the XHTML pages. We first present the user with a list of all the available comics (see Figure 6.4). When the user selects the comic of his/her choice, he/she will see the cover of the comic, together with the location(s) where the story of the comic takes place and head shots of the characters appearing in the comic (see Figure 6.5). The user can click on one of the head shots. This will send a query to the Java application, which results in a list of panels where the selected character in appears. These panels are shown to the user (see Figure 6.6). The website can be found on the CD in the folder code/website/panels.
Figure 6.2: Character-Driven Comic Search Engine with Geographical Visualization - Results shown as a grid (©WPG & Ballon Media)

Figure 6.3: Character-Driven Comic Search Engine with Geographical Visualization - Results shown on a map (©WPG & Ballon Media)
The Java application, connecting with both website, can be found in the folder `code/REST-Comic`.

### 6.2.3 Comic Information Viewer

*ComicIV (Comic Information Viewer)* is a Java application designed to view the details of an EPUB file, where the use cases from the two previous sections just focus on one aspect. After an EPUB file has been opened, the user is able to view the basic information of the comic (see Figure 6.7), which characters (see Figure 6.8) and which locations (see Figure 6.9) that appear in the story and the genres that are relevant to the comic (see Figure 6.10). For the characters and locations, the possibility exists to request more information using the *Details* button, i.e., panels/spine elements where the character/location appears in, appearance ratio regarding the panels/spine elements and the actual image files where the character/location is found. Another feature is searching for the text in the speech balloons and captions (see Figure 6.12). If more detailed or other information is needed, a custom SPARQL query can be used (see Figure 6.11; see Appendix A.3 for a number of examples). The framework used to support all this functionality is Jena (see Appendix C.2), which was also used for the Java application of the aforementioned use cases. The source code of this application can be found on the CD in the folder `code/ComicIV`.

<table>
<thead>
<tr>
<th>Comics</th>
<th>Home</th>
<th>Characters</th>
<th>Panels</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Het Eiland Amorak</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Het Gouden Paard</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Het Spreekende Testament</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lamberix</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Het Grote Gat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>De Verraderlijke Vinson</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Volle Maan</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tex en Terry</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>De Monoklende Mummie</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>De Vogels der Goden</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.4: Character-Driven Panel Extractor for Comic - List of comics
Figure 6.5: Character-Driven Panel Extractor for Comic - Selected comic ‘Het Sprekende Testament’ with list of featured characters (©WPG & Ballon Media)

Figure 6.6: Character-Driven Panel Extractor for Comic - Panels of selected character ‘Jerom’ (©WPG & Ballon Media)
Figure 6.7: ComicIV - Basic Information

Figure 6.8: ComicIV - Characters
Figure 6.9: ComicIV - Locations

Figure 6.10: ComicIV - Genres
Figure 6.11: ComicIV - SPARQL

```sparql
PREFIX dicera: <http://semweb.mmlab.be/ns/dicera#>
SELECT ?panel (STR(?name) as ?location) WHERE {
  ?panel a dicera:Panel.
  ?panel dicera:hasLayer ?layer.
  ?layer a dicera:Layer.
  ?layer dicera:hasLocation ?loc.
  ?loc a dicera:Location.
  ?loc dicera:hasName ?name.
  FILTER(langMatches(lang(?name), "nl"))
}
```

<table>
<thead>
<tr>
<th>Panel</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>panel8</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel6</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel7</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel5</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel4</td>
<td>Yawn</td>
</tr>
<tr>
<td>panel2</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel13</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel11</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel9</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel1</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel12</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel3</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel14</td>
<td>Aarde</td>
</tr>
<tr>
<td>panel10</td>
<td>Aarde</td>
</tr>
</tbody>
</table>

Figure 6.12: ComicIV - Text Search

```

<table>
<thead>
<tr>
<th>Search</th>
<th>Text</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>What are you doing here?</td>
<td>Suzy</td>
</tr>
<tr>
<td></td>
<td>Where do you come from?</td>
<td>Frank</td>
</tr>
</tbody>
</table>
```
6.3 Summary

In this chapter we discussed the implementation of a comic using the proposed solution. This implementation focused on the presentation part, however, the metadata information is available. Hence, this example can be used as a reference when studying the work done in this thesis. The use cases, centered around the metadata, display the benefits of annotating comics and using our solution. It can be used for enhancing the publisher’s website for its readers, and/or for enhancing the internal management of comics.
Chapter 7

Workflows & Authoring Environments

From meetings held, as part of the e-Strips project (mentioned in Section 1), we can conclude that the workflow followed by the publishers and authors in Belgium is of a (very) primitive nature when it comes to digitization. A lot of manual techniques are used and the digital part of their workflow is limited, i.e., limited metadata, the digital aspect is one of the last steps in their workflow, multiple languages is a problem, because of the fixed size of speech balloons, corrections are difficult, all communication is done using email, etc. In Section 7.1, we discuss the ideal workflow that should be followed to benefit the advantages of our solution. In Section 7.2, we provide an aid for transitioning from a current workflow to the ideal workflow. In Section 7.3, we discuss the conversion of old comics. In Section 7.4, we evaluate the authoring environment of the e-Strips project and we end this chapter with a summary in Section 7.5.

7.1 Ideal Workflow

In this section we will discuss the ideal workflow that has to be followed to exploit the advantages of the proposed solution of this thesis to the maximum. The order in which the actions are performed can be changed and they do not have to be executed sequentially. A number of actions can be performed together with others.

7.1.1 Creation of Content

During the creation of the content of the comic, which includes the creation of the pages, the panels and the cover, the creation of layers has to be taken into account. The original method of creating a comic consists out of drawing every panel with the result in mind.
They do not take layering in mind. The negative effect of this approach can be illustrated with the following example. In Figure 7.1 we show an example of a panel, drawn without layering. If we now want to extract the different layers for the background and characters, we would need to do this manually (through a photo editor program, for example) or we can use one of the algorithms discussed in Chapter 2. This is not a convenient way and the quality of those algorithms is more than questionable. And if we were able to dissect a character from its background, we would have a blank space on the place where the character sits, as can be seen in Figure 7.2. This makes the background useless, if it needs to be displayed before adding the characters. The method to circumvent this problem is by drawing each panel layer by layer. First, the whole background is drawn, without taking into account the places of the characters, speech balloons, etc. The same goes for the other layers. All these layers need to be digitized. This can be done by scanning the drawing or one could immediately draw its content digitally (e.g., through a tablet). The latter option is to be preferred, because scanning an image (almost) always results in quality loss.

7.1.2 Enhancement of Presentation

To enhance the presentation of the content, we use an application that relieves the authors from the burden of dealing with the technicalities of the methods discussed in Chapter 4 (the ‘Authoring Environment’ element of Figure 3.1). In what follows we will discuss how the authors will use this application, hence this can be used to define the requirements if someone would consider developing such an application\(^1\).

After all the layers are digitized (if necessary), they are loaded in the application. We group all the layers belonging to the same panel and the panels belonging to the same

---

\(^1\)We are not going to go into detail, in this thesis, about the design process of applications.
Figure 7.2: Result of dissecting a character from a single layer panel

page. This is followed with the addition of the text (if necessary) in multiple languages, together with the positioning of the text on the panel. After that the different animations (if any) for the different layers are defined for each panel. At this moment, also audio can be added. This all is finished with defining the reading order of the pages and the panels.

7.1.3 Creation of Metadata

After completing the presentation of the comic, we move on to the creation of the metadata. This can be done using the same application.

For every layer the author denotes what is present. He/she tells the application if it is dealing with a character, an object, etc. If, for example, the application is dealing with the case that it is a character, he/she connects it with the name of a character (or more specific the character object, as part of the ontology, see Section 5.2.2).

7.1.4 Generation of EPUB File

When both the necessary presentation and metadata information are present, it is now possible to generate an EPUB file of the comic. This can again be done using the application previously defined. The author should have the option to export the comic to the format defined in this thesis. Again, the user of the application is not confronted in any way with the technicalities of our solution.
7.2 Transition

In this section we will discuss how a publisher should organize the transition from its current workflow to the ideal workflow, in a realistic timeframe. A sudden turnaround of the workflow is not realistic, hence, we will transition in different parts from the old to the new workflow.

The easiest gain that the publishers can achieve is by adding metadata information to their comics, and this of course by using the solution proposed in this thesis. They do not use layering at the moment and keep it that way, however, when the pages are being digitized they add metadata, as explained in Section 7.1.3.

When they are used to the use of this metadata and their adjusted workflow, the next adjustment to their workflow handles the presentation part of the comic. The most important change here is supporting layering during the creation process, as this might be experienced as cumbersome by a number of authors. This includes the work described in Section 7.1.2.

7.3 Converting Old Comics

Next to adjusting the workflow for the creation of new comics, how do we deal with the comics that are already have been created and need to be digitized? In this section we will discuss a possible workflow for converting such comics to their digital counterpart.

We start with digitizing every page of the comic (including the cover). The second step is extracting all the panels from every page as discussed in Section 2.2. Because we are starting with single layer images (a physical comic), we can not split the panels in layers, however, this should not be a problem (see Section 4.9). The third step is the creation of the metadata, which is similar to the discussion of Section 7.1.3, only we are now dealing with only one layer, hence each layer can have multiple characters, etc. (see Section 5.4). After the addition of this information, we can continue to the generation of the EPUB file, which is completely analog to Section 7.1.4.

7.4 Authoring Environment 2.0

Authoring Environment 2.0, the authoring environment developed by Multimedia Lab for Boek.be as part of the e-Strips project (see Section 1), is a proof-of-concept created to showcase a HTML5 based workflow for creating and exporting of ebooks. After using the environment, it is clear that its main focus is textbooks. The solution proposed in

---

2http://www.boek.be
this thesis is not incorporated in this environment, hence, it is not possible to create such a comic. An additional feature that should be considered is the addition of support for our solution. The (minimal) requirements are the same as those listed in Section 7.1. This would promote the use of our solution if this environment is being used in the field. Authors, using the platform, will be able to experiment with comics in an environment where they are already accustomed to for creating textbooks. This authoring environment could ideally become the ‘Authoring Environment’ element of Figure 3.1.

7.5 Summary

In this chapter we discussed the ideal workflow when working with our solution. The workflow consists out of four steps: the creation of the content, the enhancement of the presentation, the creation of metadata and the generation of the EPUB file. We defined an application used to enhance the presentation of comics, and to add metadata. This allows us to shield authors from the technicalities of our solution. We discussed the transition from the current workflow of authors/publishers to the proposed workflow. We elaborated on the workflow to convert old comics to our solution. We reviewed the Authoring Environment developed by Multimedia Lab for Boek.be, and advised them to incorporate our solution to provide support for digital comic books.
Chapter 8

Evaluation & Discussion

In this chapter, we will evaluate if the work conducted proposes a solution for the problems, dissected from the literature, in Section 8.1. In Section 8.2, we will discuss the growing file size. In Section 8.3, the influence on the publication of comic books will be discussed. In Section 8.4, the topic is the support of our solution by several reading systems. In Section 8.5, we end with a summary of the chapter.

8.1 Literature

In this section, we discuss the different problems of the literature, as discussed in Chapter 2, and evaluate our work to determine if it offers a solution for those problems.

Content adaptation for mobile devices is not a problem anymore, you do not need to extract the different parts of a comic because everything is already separately stored in the EPUB file.

Using the proposed solution, it is possible to create a more enhanced search engine than the ones available now, when it comes to searching for descriptive information about a comic book. The use cases of Section 6.2 showcase the power of (the metadata part) of our solution and the possibility of using a SPARQL query to fulfill custom requests makes it the ideal information source for a search engine.

The creation of comic book previews using thumbnails is simplified, using our solution, because the extraction methods previously needed to dissect the panels has become superfluous.

When visualizing comics on mobile devices the biggest issue was the extraction of the visual content, so that it could be adjusted for displaying on different screen sizes. Our
solution removes the need for this extraction, because the pages are stored as a collection of panels and each panel is stored as a collection of layers. The need for adjusting the comic itself to support mobile devices can be erased completely, because our solution offers support for different screen sizes and offers reading assistance, as described in Chapter 4.

The information provided by the E-Comic Format is present in the proposed solution. The security aspect of this format is not explicitly discussed in this thesis, however, an EPUB file can use DRM, through an additional layer.

The Manga Metadata Framework, proposed in the literature, focuses on three types of information: bibliographic, structural and ontological. The bibliographic information is present, however, we do not make a difference between a (comic) entity and its publication. We see an EPUB file as a digital publication and the bibliographic information talks about that digital version of the comic. Hence, we store no information about other/print versions. Nonetheless, this information can be stored on the remote server (discussed in Section 5.3). If we consider both the information present in the EPUB file and on the remote server, we satisfy the bibliographic requirements of the framework. The information we offer, such as the story arc and issue number, provide structural information about the comic. However, it does not provide information about the relation between different story arcs, e.g., story arc A should be read before story arc B. Nonetheless, as with the bibliographic information, this information can be stored on the remote server. The ontological information is available through our use of RDFa. A final remark concerning the MMF is that it can be used together with our solution, and information stored in the MMF can be used to enhance our solution and vice versa.

To adapt comics for the visually impaired, etc., the extraction of information of comics was one of the biggest issues. However, as already discussed in this section, our solution allows to bypass these extractions.

eDBtheque tries to create a database for comic metadata information. Using our solution they do not need to extract the necessary information themselves anymore, because the information is already present in the comics. Hence, their database becomes superfluous because the goals that want to be achieved by eDBtheque can now be achieved by simply creating a database offering comic books in the format proposed by our solution.

The formats discussed in Section 2.5 all have their advantages and disadvantages. Here, we compare those formats to the proposed solution.

Our solution uses separate images for pages opposed to CBA. Administrative and descriptive metadata is available with our solution, this is not the case for CBA.
ComicBookInfo adds administrative metadata to CBA. Our solution also adds descriptive information, which is crucial for its success and adaptation. The advantage over our solution is that updating the metadata is faster, because it can be done without unzipping the archive.

CBML and ComicsML offer descriptive metadata to allow the study and analysis of comic books. The metadata is similar to the metadata of our solution. However, our information is connected to the different layers, where CMBL works on a single image.

The ACBF comes closest to the proposed solution, however, it does not support layering, which is an important requirement to circumvent the challenges from the literature.

The disadvantages of PDF, such as the limited metadata information, limited accessibility, limited adaption to the screen size, fonts, etc., limited possibilities for animations and interactivity, and limited possibility to monitor its use, are not applicable to our solution. The EPUB 3 format has the same advantage, when it comes to adaptation, as PDF, resulting in eliminating PDF’s main advantage.

The limited presentation and metadata features, when it comes to the creation of comics, of IBA, hinders its use by authors and publishers for delivering comic books to their readers. The fact that it is a proprietary format, though based on EPUB, also limits its usability to Apple devices only.

CORVIS and DigestManga are dealing with the relation between comics and movies. The problems tackled by these projects are not really related to the work conducted in this thesis. However, we encourage to use our solution to store the comics created in these projects. Adding 3D augmented reality to print comics is not applicable to the work conducted in this thesis.

8.2 EPUB File Size

When we look at the size of digital textbooks (not enhanced with multimedia) in EPUB format, we see that the file size of 162 investigated books is 813 KB on average, with a standard deviation of 1265 KB. A list of all the books can be found in Appendix D. This file size is rather small. The reason for this is obvious. Most books only contain XHTML pages without images, with the exception of the cover page. So the next question is what happens with the file size if we add the images representing the different layers?

We conducted several tests with different sizes for the images, based on the dimensions of the iPhone 5S\(^1\) and iPad\(^2\) with Retina Display. The screen resolutions are 640x1136\(^3\)

\(^1\)https://www.apple.com/iphone-5s/
\(^2\)http://www.apple.com/ipad/
\(^3\)in pixels
and 1536x2048 for the iPhone and iPad, respectively. The four dimensions, we scaled the images to, while maintaining their aspect ratio, are

- iPhone in landscape mode (image width is 640),
- iPhone in portrait mode (image height is 1136),
- iPad in landscape mode (image width is 1536),
- iPad in portrait mode (image height is 2048).

We tested this with three types of comics. The first type supports layering and uses the PNG format for storing the images. The second type uses no layering, however, it stores one PNG image for each panel. The third type differs from the second by image format used, which is JPEG. Every comic contains fifty pages, with each six panels (of the same size). If layering is enabled, we have for each panel an average of 2.67 layers and a standard deviation of 0.75 layers for the characters, one layer (and the standard deviation is zero) for the background and an average of 1.67 layers and a standard deviation of 0.47 layers for the text elements (speech balloons, captions and effects). We did not include audio into these comics and every page is represented by one XHTML page. For the non-layered versions of the comic, we merged all the different layers into one image.

Aside from the aforementioned tests where we maintain the aspect ratio of the images, we also tested what happens if both dimensions are scaled to the defined size, i.e., iPhone in landscape mode means scaling the image to 640x640, regardless of the original ratio. This gives us three additional results (tagged with ‘square’), which are presented, together with the first three results, in Figure 8.1. We see that the sizes increase, when looking at the three results last mentioned. This is because when the ratio of the images is increased, the file size follows. The first three result show an increase in file size when increasing the ratio, with exception of the iPad in landscape mode. The cause is the original resolution of the panels which is 600x400. When we scale them to a ratio with a width of 1536 (iPad in landscape mode), it results in 1536x1024. When we scale them to a ratio with a height of 1136 (iPhone in portrait mode), it results in 1704x1136. We can see that the resolution is bigger for the iPhone than for the iPad. The conclusion is that the file size will depend on resolution of the panels and the resolution of the device.

Next to our self-created comic, we also have a real comic from the ‘Suske en Wiske’ series (which is not layered). Here, we are dealing with 539 panels, which is much more than the 300 (= 50 · 6) from our comic. This is because they use different and smaller sizes for their panels, hence, the increase in the number of panel per page. However, the average total amount of pages is 50, which allows us to compare the two comics. In Figure 8.2 the graph with the results can be found. In all cases the ‘Suske en Wiske’ comic results
in a bigger file size. The main reason for this can be the way the different images are created. Our comic is based on vector images and we use only solid colors. The ‘Suske en Wiske’ comic on the other hand is drawn by hand and is scanned in afterwards, hence we are not dealing with solid colors. We tested if this is true or not. Adding 5-6% noise (which can be the result of scanning and using non-solid colors) to our panels results in the same file size for the panels of the same dimensions of the ‘Suske en Wiske’ comic. Of course take into account that different applications possibly use different compression algorithms and results might differ when trying to reconstruct these results.

Textbooks vs. comics Comparing the size of our comic to the sizes of textbooks in EPUB format, as mentioned in the beginning of this section, shows us that the file sizes of a comic designed for the iPhone in landscape mode and portrait mode are 10,000 times and 20,000 times bigger, respectively, than an average textbook. Of course the main (and only) reason for this is the addition of image files to the comic.

Different devices From Figures 8.1 and 8.2, we conclude that the size of the EPUB files grows if the screen resolution grows. Should a publisher then offer different files with different resolution for the images? The publisher is able to do that or he/she might opt for one of the large sizes, because the images can be downscaled if necessary. This approach, of course, leads to larger file sizes, however, zooming in on the images results in more detail.

ComiXology’s new app ComiXology announced the introduction of a new mobile application for reading their comics, on Android and iOS. Users must synchronize the content of the old app to their comiXology account. Logging into the new application offers the possibility to download the comics synchronized to their account. For a user named Mike this results in a download of 26 GB of comics. This simple example shows that the file size of digital comics should not be neglected.

8.3 Distribution Channels

The problem statement (see Section 1.1) mentions the use of different (proprietary) formats by the different distribution channels causing additional work for publishers and authors to make their work available on these platforms. From the work conducted in this thesis, we conclude that the features offered by those different platforms are replicable.

\[^4\text{using Pixelmator (version 2.1.2; http://www.pixelmator.com)}\]

\[^5\text{http://thenextweb.com/apps/2014/04/26/comixology-removes-app-purchases-acquired- amazon/comment-1357269754}\]
Figure 8.1: Graph comparing file size different types of comics; iPhone-L stands for iPhone in landscape mode, iPhone-P for iPhone in portrait mode, iPad-L for iPad in landscape mode and iPad-P for iPad in portrait mode.

Figure 8.2: Graph comparing the file size of custom comic and ‘Suske en Wiske’ comic; iPhone-L stands for iPhone in landscape mode, iPhone-P for iPhone in portrait mode, iPad-L for iPad in landscape mode and iPad-P for iPad in portrait mode.
using our proposed solution. Hence, we advice the companies behind these platforms to revise our solution, and eventually replace their current format by ours. This eases the publication of content for authors to their channel, and a standardized format is used that is actively being developed. A point of criticism might come from big publishers saying that they need their own dedicated application to offer a special and unique experience to their readers. This, however, is not valid. The presentation functionality does not depend on the reading system. It is embedded in the EPUB file itself, allowing it to be read on every system supporting the EPUB 3 specification. It also allows them to discontinue the development of their custom application for reading comics or to work towards a reading system supporting the (complete) EPUB 3 specification, for more information about this topic we refer to Section 9.9.

8.4 Reading System Support

We have created an EPUB file that is conform the specifications of EPUB 3, however, how well does it work when it is actually being read? We have tested the EPUB file of Section 6.1 in three applications: Readium, Calibre and iBooks, and on one device: the Kobo Glo\textsuperscript{6}. In what follows we discuss how good they were able to handle our EPUB file, considering that they (claim to) implement the EPUB 3 specification.

8.4.1 Readium

The Readium Foundation\textsuperscript{7}, concerned with accelerating the adaptation of the EPUB format, offers a Chrome extension to manage and read (the books in an) EPUB library. The current version, version 2.13.4, broke the presentation of our solution, while it worked in the previous version. This forced us to rollback to an older version, version 0.9. This was the most recent compiled version that could be found online (with exception of the current version). This older version allowed us the confirm again that our comic book works in Readium. The only disadvantage is that when zooming out to view the whole page, the page is at 100% its size, however, this does not mean necessarily that the reader can view the whole page. The zoom library, introduced in Section 4.4, could be altered to support this behavior. Another subject of study is why the current version of Readium does not allow us to read the comic. At the moment, it shows only a part of each panel.

\textsuperscript{6}http://www.kobo.com/koboglo
\textsuperscript{7}http://readium.org/
8.4.2 Calibre

We used Calibre in Section 3.2.1 to test the research efforts towards the use of the EPUB 3 format for comics, described in Section 2.5. Using it to test the proposed solution, gives the following result. We are presented with the first panel. Scripting is enabled and we can show the different layers. Navigating to the next panel is not possible. The reason might be that (a part of) the scripting functionality, required by the libraries we use, is not available. If we use the controls provided by Calibre, however, we can navigate to the next panel. This works only for the first panel. Navigating to the third panel (from the second), results in navigating to the second page. Another problem is that a number of the fallback panels are also being displayed, however, scripting is enabled, so they should be hidden. Disabling scripting in the EPUB file itself, forces Calibre to only show the fallback panels (see Section 4.8). Navigating, however, remains a problem. From our test, it seems that Calibre navigates through the page from the left to the right, regardless of the content at the bottom of the page (the two last panels) that is not being displayed (completely).

8.4.3 iBooks

iBooks\(^8\) (version 3.2) supports the EPUB format, however, this could not be proven by using our EPUB file. Although we are using only technologies defined by the EPUB 3 specification, iBooks assumes that the user is reading a book and not necessary a comic. We come to this conclusion because of the following. When the user wants to display the next layer by touching the screen the tool bars of iBooks appear. The only gestures he/she can use, regarding navigating through the content, is touching the side of the screen or swiping to the desired direction (cfr. a physical book). When the user does this in our case he/she moves to the next page, hence, he/she does not see the next layer or the next panel of that page. The real problem here is that iBooks captures all physical input (e.g., gestures) and executes its own code. Our JavaScript code for navigating through the comic is being ignored completely, making it impossible to read the comic. Disabling JavaScript, causing iBooks to display the fallback panels, results in showing one panel for each page. The other panels are neglected when navigating to the next page.

8.4.4 Kobo Glo

We have tested our solution on a Kobo Glo running version 2.5.2 of its software. When the EPUB file was loaded by the eReader it became clear that the needed scripting func-
tionality is not supported. EPUBTest⁹, a website focusing on testing and summarizing reading system’s their support of various EPUB 3 features, gives the Kobo eReaders a rating of 27.1% for the support of scripting (for version 3.1.1 of the software). However, this is not a major issue that makes our comic unreadable. We discussed in Section 4.8 what happens when scripting is disabled or not available. The result is illustrated when using the Kobo Glo. For every (scripted) panel we see the fallback panel instead. We are presented with two consecutive panels. When navigating to the next page (on the Kobo, this is not a comic book page), we see the next two panels, etc., as can be seen on Figure 8.3.

Figure 8.3: Proposed solution on Kobo Glo, using fallback panels

⁹http://www.epubtest.org
8.5 Summary

In this chapter we discussed and evaluated the proposed solution. We concluded that most of the problems investigated by several research efforts are solved when using our solution. The increasing file size when storing graphical information in an EPUB file needs to be considered, when opting to use our solution for distributing comic books to several types of devices. The overhead of distributing comic books through different channels can be removed, if those channels use our solution, which uses a standard that is being actively developed. Readium (with the use of scripting) and the Kobo Glo (using the fallback panels) allow us to read the comic. iBooks and Calibre are not able to display the comic in a correct way, both with or without scripting.
Chapter 9

Future Work

In this chapter we will discuss the future work that needs to be done regarding digital comics, and how this leads to the ‘comic of the future’. In Section 9.1, we talk about the linking with the other multimedia content. In Section 9.2, we discuss the information of the remote server. In Section 9.3, we look at how our solution can aid the recommendation of comics to readers. In Section 9.4, we discuss the implications of different types of comics. In Section 9.5, we talk about the possible optimizations. In Section 9.6, we discuss the introduction of a layered cover. In Section 9.7, we talk about the possible extensions to our ontology. In Section 9.8, we discuss the incorporation of the new EPUB specifications. In Section 9.9, the topic is the development of a reading system supporting our solution. In Section 9.10, we talk about the comic of the future, and we end this chapter with a summary in Section 9.11.

9.1 Linking with Other Multimedia

As we mentioned in Chapter 1, the comic industry is not limited to comic books. It stretches far beyond these boundaries. When talking about comics we also need to include the movies and video games. Stories that first have been told on paper made (or are making) their way to the (big) screen. Now that we are dealing with digital comics, digital movies and digital video games can we link these different types of multimedia?

Our solution includes global identifiers in the metadata. If a digital movie would be able to link their character to these global identifiers then, for example, we would be able to search all movies that have the same characters as a specific comic. The same goes for video games.

Synchronization between different types of multimedia is also a possible topic of research, which is also the topic of Jeppe Knockaert’s thesis called Alignment of Text and Video: Matching Ebooks and Their Film Adaptations. After reading a few pages in a
comic, a reader might like to continue the story on his/her TV screen. How can we realize this mapping between the panels in a comic and the frames of a movie?

These are only two use cases of many of the linking of different multimedia types, and show that it is an interesting topic of study for the future.

9.2 Remote Server Information

In Section 5.3 we discussed how we can connect the local and remote information. This approach uses a remote server. The question that rises here is how the reading system knows the necessary information about how to contact that server. Is this information stored in the EPUB file or maintains the reading system a list with all the servers of all the known publishers? Because updating an EPUB file is not easy (at the moment) we suggest the second option. The reading system can update its list when needed, so it always connects to the correct server with the correct parameters. The topic for future work is clear and includes studying the best method for obtaining and maintaining the remote server’s information.

9.3 Recommendation

Most of today’s publishers have an online website were users can browse their comic collection. This enables them to discover new comics. Sometimes this is based on the comic books they previous read. The website will recommend them comics from the same story arc, for example.

When the comics are stored using the solution proposed in this thesis, one could enhance the recommendation system of such a website. Consider the following situation where a reader has already read 25 comics of several story arcs, however, analyzing these comics more closely (using the metadata information), we can conclude that he/she prefers to read comics that are centered around the character ‘Wolverine’. We know this because Wolverine is present on at least 70% of the panels of all the comic books he/she read. Now, if the publisher incorporates a use case similar to the one described in Section 6.2.1 on his website, he will be able to use the presence information (of 70%) to search for comics that include Wolverine and he can display them to the reader (in a visual way).

Future work on this topic consists out of studying which elements of the metadata information part can be used to enhance the recommendation system of publishers to drive the discovery of new comics of their readers.
9.4 Different Types of Comics

The examples shown in this thesis are all flat comics (with a number of small exceptions for the ‘Suske en Wiske’ comics, i.e., where a number of speech balloons go out of the panels). The question that rises is: does our solution also work for non-flat comics. From the literature (see Section 2.1), we know that the extraction of panels from non-flat comics is more difficult than for flat comics. Of course, our solution bypasses the need for such an extraction step (with exception of Section 7.3), however, are we able to display and annotate such a type of comics?

This could be the topic of future work, however, we can already suggest from the work conducted in this thesis that the functionality offered by the EPUB format (and more specific the OWF) is more than powerful enough to fulfill the additional requirements set by non-flat comics. This does not guarantee that solution stays as simple as it is for flat comics. A non-exhaustive summary of requirements can be found in the following list.

- Background of a page is not blank, however, it can be an image.
- The presentation of content is not restricted to the borders of a panel.
- Content outside a panel needs to be annotated.

9.5 Optimization

In the work conducted we rely on external JavaScript libraries such as jQuery and Zoomooz. What we did not do in this thesis, as explained in Section 4.4, is comparing these libraries to other libraries offering the same functionality, because we only wanted to prove that the required functionality can be provided by JavaScript. The topic of future work can be comparing these libraries based on their functionality and efficiency.

A possible alternative for the Zoomooz library could be the library written by Hakim El Hattab\textsuperscript{1,2}.

The use of jQuery might not be necessary for a number of functionalities needed in our examples. A number of functions provided by this plugin are also possible without it, such as explained on the website with the name ‘You Might Not Need jQuery’\textsuperscript{3}.

As explained in Section 4.7, we bundled the functionality that can be useful for other people who want to create a digital comic book using our solution. In that section we already pointed at a design issue that needs to be studied. The remark that we can make

\textsuperscript{1}\url{http://hakim.se/projects/zoom-js}
\textsuperscript{2}We did not test this library.
\textsuperscript{3}\url{http://youmightnotneedjquery.com}
is that, again, the implementation of the functionality provided here might not be optimal and could be the topic of future work.

9.6 Layering the Cover

In this thesis we focussed on the actual content of a comic: the pages and panels. Besides these two elements, we also have the cover of a comic book. The topic of future work can be how we need to expand the solution of this thesis to incorporate layering and metadata for a cover. The ontology might need to be adjusted with, for example, the addition of a cover page entity. The functionality, with regard to presentation, is already present in the current solution and can be used in the same way for the cover.

9.7 Extending the Ontology

Dicera, the ontology designed in this thesis, has been compared to Schema.org and FOAF (more information about these two projects in Section 5.2.2). This information can be used to extend or complete the ontology. Besides the additional need for the support of layered covers, see Section 9.6, we could also add additional concepts to the ontology. One such a concept can be the concept of teams, such as ‘The Avengers’ and ‘X-Men’. We can denote in which comics they appear, similar to the concept of a character, and we can specify which character belongs to which team.

9.8 EPUB Specifications

In Sections 4.10 and 4.11, we discussed two specifications useful for comic books and for our proposed solution. Future work might consist out of the elaborating on these specifications and providing a adapted solution, based on our solution. The consideration should be made to comment on these specifications, with the goal to drive the development towards the solution proposed in this thesis, because the specifications lack some features that our solution offers, and that we consider necessary for a digital comic.

9.9 Reading System Support

In Section 8.4, we tested our solution on several devices and applications. We came to the conclusion that a number of these applications, claiming to support EPUB 3, are not able to display the comic in a desirable or readable way to the user. The work conducted
in this field should, hence, consist out the development of an application that supports the features of the EPUB 3 specification needed for the use of the proposed solution.

9.10 Comic of the Future

How should the comic of the future look like? What format will it use? How does it relate to the other types of multimedia?

The ultimate digital comic would use only one format for all authors, publishers, applications and devices. We choose the EPUB 3 format in this thesis and believe that it is the perfect vehicle for the comic of the future (see Section 3.2.1).

The comic of the future should not limit itself to comics, however, it should connect to other multimedia content. This can be the synchronization of a comic and its movie counterpart or a comic and its book counterpart. Not only the synchronization of content is an option, however, also the linking of all the multimedia related to comics.

The interaction between the reader and the comic can be enhanced, i.e., the addition of gamification to make a comic more engaging, which is the topic of Laurens De la Marche’s thesis called *Feasibility and Usability of a Generic Gamification Framework in Digital Books*. For example, a reader can unlock additional content if he/she reads a comic on a specific location. The location information is present in the comic and (almost) every smartphone or tablet is able to do localization of the user.

9.11 Summary

The topics discussed in this chapter show that a lot of work has to be done in the field of digital comics: both in the direction of the enhancement of the proposed solution and the creation of applications, reading systems and frameworks that work with this format to enhance its adaptation. Enhancing the solution is done by investigating the support of different types of comics, optimizing and extending the presentation features, extending the ontology, studying the use of the new (draft) specifications for EPUB 3, and studying how it can be used to recommend new comics to readers. Work needs to be conducted to create a remote server to work with our solution and a framework that links other multimedia such as movies to comic books, and to develop a reading system that fully supports our solution. We finished this chapter with the discussing of the comic of the future, which according to us, should be based on our solution, should incorporate more linking between other multimedia and should offer (more) user interaction.
Chapter 10

Conclusion

In this thesis, we offer a solution for the problems studied in the literature, and the problems caused by the use of different (proprietary) formats for the different publication platforms.

The proposed solution consists out of a **presentation** part and **metadata** part, and builds on the **EPUB 3 format**. The presentation part offers animations, reading assistance, audio and multiple languages, through the introduction of layering and the use of **Open Web Formats**. The availability of scripting is an important requirement of our solution, however, when a reading system lacks this feature, a fallback procedure is foreseen. Research towards the enhancement of the EPUB 3 specification, when it comes to comic books, has been conducted, and can be possibly used together with our solution. The metadata part enhances the administrative metadata of EPUB 3, and adds descriptive information. This is done through the use of both local information in the EPUB file and remote information on, for example, a remote server. Both types of information can be linked, which enables reading systems to request more information based on the metadata available in the EPUB file. To structure the metadata, we designed an ontology called **Dicera**. It is available in both RDF and OWL format, allowing it to be used together with **RDFa**, which is recently added to the EPUB 3 specification.

The introduction of a new (or updated) format might mean a change in the **workflow of creating comic books**. In this thesis, we discussed the ideal workflow, when using our solution. It consists out of creating the content, enhancing the presentation, creating the metadata and generating the EPUB file.

Evaluation of the proposed solution, against the problems denoted at the beginning of this chapter, shows us that it **solves most of the problems** and should be considered as the default format for distributing digital comics. However, the increasing file size, and the sometimes limited EPUB 3 support of reading systems points us to possible problems, when deploying this solution.
A lot of topics can be studied, in the future, in the field of digital comics. We talked about the linking between other multimedia, recommending comics to readers using the metadata, the support of different types of comics, the optimization of the implementation of the solution, the incorporation of the new EPUB 3 specifications, and increasing the reading system support. All this is done to work towards a comic book of the future, which is subconsciously the goal of this thesis.
Appendix A

Ontologies

A.1 RDF

Listing A.1: Dicera - RDF Schema

```xml
<?xml version='1.0'?>

<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
    xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
    xmlns='http://semweb.mmlab.be/ns/dicera#'
    xml:base='http://semweb.mmlab.be/ns/dicera#'>

  <rdfs:Class rdf:ID='Comic'>
    <rdfs:comment>This class represents a comic entity.</rdfs:comment>
  </rdfs:Class>

  <rdfs:Class rdf:ID='StoryArc'>
    <rdfs:subClassOf rdfs:resource='#ComicContentElement'/>
    <rdfs:comment>This class represents a story arc entity.</rdfs:comment>
  </rdfs:Class>

  <rdfs:Class rdf:ID='Genre'>
    <rdfs:subClassOf rdfs:resource='#ComicContentElement'/>
    <rdfs:comment>This class represents a genre entity.</rdfs:comment>
  </rdfs:Class>

  <rdfs:Class rdf:ID='ContentRating'>
    <rdfs:comment>This class represents a content rating entity</rdfs:comment>
  </rdfs:Class>

</rdf:RDF>
```

85
<rdfs:Class rdf:ID="ComicLayoutElement">  
  <rdfs:comment>This class represents all the elements of a comic that add to the layout.</rdfs:comment> 
</rdfs:Class>

<rdfs:Class rdf:ID="ComicContentElement">  
  <rdfs:comment>This class represents all the elements of a comic that add to or tell about the content.</rdfs:comment> 
</rdfs:Class>

<rdfs:Class rdf:ID="Character">  
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>  
  <rdfs:comment>This class represents a character entity.</rdfs:comment> 
</rdfs:Class>

<rdfs:Class rdf:ID="Issue">  
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>  
  <rdfs:comment>This class is used to denote the issue number of comic and to connect it to a story arc.</rdfs:comment> 
</rdfs:Class>

<rdf:Property rdf:ID="isOnCover">  
  <rdfs:domain rdf:resource="#Character"/>  
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>  
  <rdfs:comment>This boolean value denotes whether a character is on the cover of the comic or not. If this property is not defined, we assume the value false.</rdfs:comment> 
</rdf:Property>

<rdfs:Class rdf:ID="ComicTextElement">  
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>  
  <rdfs:comment>This class represents all the elements of a comic that represent text.</rdfs:comment> 
</rdfs:Class>

<rdfs:Class rdf:ID="Balloon">  
  <rdfs:subClassOf rdf:resource="#ComicTextElement"/>  
  <rdfs:comment>This class represents a speech balloon in a comic.</rdfs:comment> 
</rdfs:Class>
<rdfs:Class rdf:ID=""Caption"">  
  <rdfs:subClassOf rdf:resource="#ComicTextElement"/>  
  <rdfs:comment>This class represents a caption in a comic.</rdfs:comment>  
</rdfs:Class>

<rdfs:Class rdf:ID=""Effect"">  
  <rdfs:subClassOf rdf:resource="#ComicTextElement"/>  
  <rdfs:comment>This class represents an effect (i.e. 'Zzzz', 'Boem') in a comic.</rdfs:comment>  
</rdfs:Class>

<rdfs:Class rdf:ID=""Location"">  
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>  
  <rdfs:comment>This class represents a location, i.e. a place where a scene (or the whole comics) takes place.</rdfs:comment>  
</rdfs:Class>

<rdfs:Class rdf:ID=""Object"">  
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>  
  <rdfs:comment>This class represents a (special) object, i.e. a treasure that returns in multiple comics.</rdfs:comment>  
</rdfs:Class>

<rdfs:Class rdf:ID=""Text"">  
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>  
  <rdfs:comment>This class represents the text that is visually presented to the reader.</rdfs:comment>  
</rdfs:Class>

<rdfs:Class rdf:ID=""Audio"">  
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>  
  <rdfs:comment>This class represents audio that can be played to the reader i.e. using Javascript.</rdfs:comment>  
</rdfs:Class>

<rdfs:Class rdf:ID=""Panel"">  
  <rdfs:subClassOf rdf:resource="#ComicLayoutElement"/>  
  <rdfs:comment>This class represents a panel on the page of a comic.</rdfs:comment>  
</rdfs:Class>

<rdfs:Class rdf:ID=""Page"">  
  <rdfs:subClassOf rdf:resource="#ComicLayoutElement"/>  
</rdfs:Class>
<rdfs:comment>This class represents the page of a comic.</rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="Layer">
  <rdfs:subClassOf rdf:resource="#ComicLayoutElement"/>
  <rdfs:comment>This class represents the layer of panel, each layer can hold multiple characters, objects etc.</rdfs:comment>
</rdfs:Class>

<!--
  Tells if a comic talks about a certain character.
  It is not because a character is defined that the comic talks about it, hence a application using this information should only assume that a certain character appears in the comic if it is linked to the comic using this property.
-->

<rdf:Property rdf:ID="talksAboutCharacter">
  <rdfs:domain rdf:resource="#Comic"/>
  <rdfs:range rdf:resource="#Character"/>
  <rdfs:comment>This property connects character entities to a comic entity.</rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="withEmotion">
  <rdfs:domain rdf:resource="#Text"/>
  <rdfs:comment>This property tells what emotion the text expresses.</rdfs:comment>
</rdf:Property>

<!-- Because effects are special images, one can not assume that the built-in methods (HTML5, CSS) are used to create this images and text. Hence if the text is imbedded in the image, we use this property to denote what the text of the effect is.
-->

<rdf:Property rdf:ID="hasLexicalForm">
  <rdfs:domain rdf:resource="#Effect"/>
  <rdfs:comment>The lexical form of an effect, for use if the effect is created by an image file with embedded text.</rdfs:comment>
</rdf:Property>

<!-- Tells whether a comic has (a) defined location(a) or not, same reason as for talksAboutCharacter -->
<rdf:Property rdf:ID="'talksAboutLocation'">
  <rdfs:domain rdf:resource="#Comic"/>
  <rdfs:range rdf:resource="#Location"/>
  <rdfs:comment>This property connects location entities to a comic entity.</rdfs:comment>
</rdf:Property>

<!— Tells whether a comic has (a) defined object (s) or not, same reason as for talksAboutCharacter —>
<brdf:Property rdf:ID="'talksAboutObject'">
  <rdfs:domain rdf:resource="#Comic"/>
  <rdfs:range rdf:resource="#Object"/>
  <rdfs:comment>This property connects location entities to a comic entity.</rdfs:comment>
</rdf:Property>

<!— Tells whether a comic has (a) defined genre (s) or not, same reason as for talksAboutCharacter —>
<brdf:Property rdf:ID="'hasGenre'">
  <rdfs:domain rdf:resource="#Comic"/>
  <rdfs:range rdf:resource="#Genre"/>
  <rdfs:comment>This property connects genre entities to a comic entity.</rdfs:comment>
</rdf:Property>

<!— Tells the order in which a comicLayoutElement (mainly intended for panels and pages) has to be read, in relation to the other panels. We reset the counter for the panels for every new page. —>
<brdf:Property rdf:ID="'readingOrder'">
  <rdfs:domain rdf:resource="#ComicLayoutElement"/>
  <rdfs:comment>The order in which a ComicLayoutElement has to be read.</rdfs:comment>
</rdf:Property>

<!— Tells whether a comic has a defined content rating or not, same reason as for talksAboutCharacter —>
<brdf:Property rdf:ID="'hasContentRating'">
  <rdfs:domain rdf:resource="#Comic"/>
  <rdfs:range rdf:resource="#ContentRating"/>
  <rdfs:comment>This property connects content rating entities to a comic entity.</rdfs:comment>
</rdf:Property>

<brdf:Property rdf:ID="'usesSystem'">
  <rdfs:domain rdf:resource="#ContentRating"/>
</rdf:Property>
This property tells what content rating system a certain content rating entity uses.

This property tells what rating a certain content rating has for the comic.

This property tells how many percent of the genre actually matches with the story of the comic.

This property connects story arc entities to a issue.

This property connects a comic with an issue.

This property tells what the issue number is of the issue.

This property connects a ComicLayoutElement to a comic.

This property connects a ComicContentElement to a ComicLayoutElement.
<rdfs:range rdf:resource="#ComicContentElement"/>
<rdfs:comment>This property connects a ComicContentElement to a comic.</rdfs:comment>

210</rdf:Property>

<rdf:Property rdf:ID="hasLayer">
  <rdfs:domain rdf:resource="#Panel"/>
  <rdfs:range rdf:resource="#Layer"/>
  <rdfs:comment>This property makes a layer part of a panel.</rdfs:comment>
</rdf:Property>

<rdfs:domain rdf:resource="#Panel"/>
<rdfs:range rdf:resource="#Layer"/>

215</rdfs:comment>This property makes a layer part of a panel.</rdfs:comment>

</rdf:Property>

<rdfs:domain rdf:resource="#Page"/>
<rdfs:range rdf:resource="#Panel"/>

220</rdfs:comment>This property makes a panel part of a page.</rdfs:comment>

</rdf:Property>

<rdfs:domain rdf:resource="#Comic"/>
<rdfs:range rdf:resource="#Page"/>

225</rdfs:comment>This property makes a page part of a comic.</rdfs:comment>

</rdf:Property>

<rdfs:domain rdf:resource="#Layer"/>
<rdfs:range rdf:resource="#Location"/>

230</rdfs:comment>This property tells if the scene of a certain panel is located on a certain location.</rdfs:comment>

</rdf:Property>

<rdfs:domain rdf:resource="#Layer"/>
<rdfs:range rdf:resource="#Character"/>

235</rdfs:comment>This property tells if a layer represents a certain character, 1 layer can represent multiple characters.</rdfs:comment>

</rdf:Property>

<rdfs:domain rdf:resource="#Layer"/>
<rdfs:range rdf:resource="#Object"/>

240</rdfs:comment>This property tells if a layer represents a certain object, 1 layer can represent multiple objects.</rdfs:comment>

</rdf:Property>
<rdf:Property rdf:ID="hasAudio">
  <rdfs:domain rdf:resource="#Layer"/>
  <rdfs:range rdf:resource="#Audio"/>
  <rdfs:comment>This property tells if a layer represents audio, 1 layer can represent multiple audio objects.</rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="hasTextElement">
  <rdfs:domain rdf:resource="#Layer"/>
  <rdfs:range rdf:resource="#ComicTextElement"/>
  <rdfs:comment>This property tells if a layer represents a certain ComicTextElement, 1 layer can represent multiple ComicTextElement.</rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="hasText">
  <rdfs:domain rdf:resource="#Layer"/>
  <rdfs:range rdf:resource="#Text"/>
  <rdfs:comment>This property tells if a layer displays text.</rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="says">
  <rdfs:domain rdf:resource="#Text"/>
  <rdfs:comment>This property denotes the actual text that is being said by a Text class.</rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="isBalloonFrom">
  <rdfs:domain rdf:resource="#Balloon"/>
  <rdfs:range rdf:resource="#Character"/>
  <rdfs:comment>This property tells to which character a certain speech balloon belongs.</rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="isSaidThrough">
  <rdfs:domain rdf:resource="#Text"/>
  <rdfs:range rdf:resource="#ComicTextElement"/>
  <rdfs:comment>This property tells through which ComicTextElement the text inside a Text class is being said.</rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="hasName"/>
This class represents the name of a ComicContentElement.

This class represents the global identifier of a ComicContentElement, i.e. for remote lookup of extra information.

### A.2 OWL

Listing A.2: Dicera - OWL Schema

```xml
<?xml version='1.0'?>

<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
         xmlns:owl='http://www.w3.org/2002/07/owl#'
         xmlns='http://semweb.mmlab.be/ns/dicera#'
         xml:base='http://semweb.mmlab.be/ns/dicera#'>
  <owl:Ontology rdf:about=''>
    <dc:description>A vocabulary to describe (digital) comic books, as part of the thesis ‘Enhanced Presentation and Machine–Understandable Metadata for Digital Comics using Open Web Formats’.</dc:description>
  </owl:Ontology>

  <owl:Class rdf:about='Comic'>
    <rdfs:comment>This class represents a comic entity.</rdfs:comment>
  </owl:Class>

  <owl:Class rdf:about='StoryArc'>
    <rdfs:subClassOf rdf:resource='#ComicContentElement'/>
    <rdfs:comment>This class represents a story arc entity.</rdfs:comment>
  </owl:Class>

  <owl:Class rdf:about='Genre'>
    <rdfs:subClassOf rdf:resource='#ComicContentElement'/>
  </owl:Class>
</rdf:RDF>
```
<rdfs:comment>This class represents a genre entity.</rdfs:comment>
</owl:Class>

<owl:Class rdf:about=""ContentRating"">
  <rdfs:comment>This class represents a content rating entity.</rdfs:comment>
</owl:Class>

<owl:Class rdf:about=""ComicLayoutElement"">
  <rdfs:comment>This class represents all the elements of a comic that add to the layout.</rdfs:comment>
</owl:Class>

<owl:Class rdf:about=""ComicContentElement"">
  <rdfs:comment>This class represents all the elements of a comic that add to or tell about the content.</rdfs:comment>
</owl:Class>

<owl:Class rdf:about=""Character"">
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>
  <rdfs:comment>This class represents a character entity.</rdfs:comment>
</owl:Class>

<owl:Class rdf:about=""Issue"">
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>
  <rdfs:comment>This class is used to denote the issue number of comic and to connect it to a story arc.</rdfs:comment>
</owl:Class>

<owl:ObjectProperty rdf:ID=""isOnCover"">
  <rdfs:domain rdf:resource="#Character"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdfs:comment>This boolean value denotes whether a character is on the cover of the comic or not. If this property is not defined, we assume the value false.</rdfs:comment>
</owl:ObjectProperty>

<owl:Class rdf:about=""ComicTextElement"">
  <rdfs:subClassOf rdf:resource="#ComicContentElement"/>
  <rdfs:comment>This class represents all the elements of a comic that represent text.</rdfs:comment>
</owl:Class>
This class represents a speech balloon in a comic.

This class represents a caption in a comic.

This class represents an effect (i.e. 'Zzzz', 'Boem') in a comic.

This class represent a location, i.e. a place where a scene (or the whole comic) takes place.

This class represent a (special) object, i.e. a treasure that returns in multiple comics.

This class represents the text that is visually presented to the reader.

This class represents audio that can be played to the reader i.e. using Javascript.
<owl:Class rdf:about="Panel"/>
   <rdfs:subClassOf rdf:resource="#ComicLayoutElement"/>
   <rdfs:comment>This class represents a panel on the page of a comic.</rdfs:comment>
</owl:Class>

<owl:Class rdf:about="Page"/>
   <rdfs:subClassOf rdf:resource="#ComicLayoutElement"/>
   <rdfs:comment>This class represents the page of a comic.</rdfs:comment>
</owl:Class>

<owl:Class rdf:about="Layer"/>
   <rdfs:subClassOf rdf:resource="#ComicLayoutElement"/>
   <rdfs:comment>This class represents the layer of panel, each layer can hold multiple characters, objects etc.</rdfs:comment>
</owl:Class>

<owl:ObjectProperty rdf:ID="talksAboutCharacter"/>
   <rdfs:domain rdf:resource="#Comic"/>
   <rdfs:range rdf:resource="#Character"/>
   <rdfs:comment>This property connects character entities to a comic entity.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="withEmotion"/>
   <rdfs:domain rdf:resource="#Text"/>
   <rdfs:comment>This property tells what emotion the text expresses.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasLexicalForm"/>

<!-- Tells if a comic talks about a certain character. It is not because a character is defined that the comic talks about it, hence a application using this information should only assume that a certain character appears in the comic if it is linked to the comic using this property. -->

<!-- Because effects are special images, one can not assume that the built-in methods (HTML5, CSS) are used to create this images and text. Hence if the text is imbedded in the image, we use this property to denote what the text of the effect is. -->

<!--
<owl:ObjectProperty rdf:ID="hasLexicalForm"
<owl:ObjectProperty rdf:ID="withEmotion"
<owl:ObjectProperty rdf:ID="talksAboutCharacter"
<owl:Class rdf:about="Panel"
<owl:Class rdf:about="Page"
<owl:Class rdf:about="Layer"

96
<rdfs:domain rdf:resource="#Effect"/>
<rdfs:comment>The lexical form of an effect, for use if the effect is created by an image file with embedded text.</rdfs:comment>
</owl:ObjectProperty>

<!— Tells whether a comic has (a) defined location(s) or not, same reason as for talksAboutCharacter —>
<owl:ObjectProperty rdf:ID="talksAboutLocation"/>
<rdfs:domain rdf:resource="#Comic"/>
<rdfs:range rdf:resource="#Location"/>
<rdfs:comment>This property connects location entities to a comic entity.</rdfs:comment>
</owl:ObjectProperty>

<!— Tells whether a comic has (a) defined object(s) or not, same reason as for talksAboutCharacter —>
<owl:ObjectProperty rdf:ID="talksAboutObject"/>
<rdfs:domain rdf:resource="#Comic"/>
<rdfs:range rdf:resource="#Object"/>
<rdfs:comment>This property connects location entities to a comic entity.</rdfs:comment>
</owl:ObjectProperty>

<!— Tells whether a comic has (a) defined genre(s) or not, same reason as for talksAboutCharacter —>
<owl:ObjectProperty rdf:ID="hasGenre"/>
<rdfs:domain rdf:resource="#Comic"/>
<rdfs:range rdf:resource="#Genre"/>
<rdfs:comment>This property connects genre entities to a comic entity.</rdfs:comment>
</owl:ObjectProperty>

<!— Tells the order in which a comicLayoutElement (mainly intended for panels and pages) has to be read, in relation to the other panels.
We reset the counter for the panels for every new page. —>
<owl:ObjectProperty rdf:ID="readingOrder"/>
<rdfs:domain rdf:resource="#ComicLayoutElement"/>
<rdfs:comment>The order in which a ComicLayoutElement has to be read.</rdfs:comment>
</owl:ObjectProperty>

<!— Tells whether a comic has a defined content rating or not, same reason as for talksAboutCharacter —>
<owl:ObjectProperty rdf:ID="hasContentRating"/>
This property connects content rating entities to a comic entity.

This property tells what content rating system a certain content rating entity uses.

This property tells what rating a certain content rating has for the comic.

This property tells how many percent of the genre actually matches with the story of the comic.

This property connects story arc entities to an issue.

This property connects a comic with an issue.

This property tells what the issue number is of the issue.

This property connects story arc entities to an issue.
<rdfs:domain rdf:resource="#Comic' '/>
<rdfs:range rdf:resource="#ComicLayoutElement' '/>
<rdfs:comment>This property connects a ComicLayoutElement to a comic.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasComicContentElement' '"
<rdfs:domain rdf:resource="#ComicLayoutElement' '/>
<rdfs:range rdf:resource="#ComicContentElement' '/>
<rdfs:comment>This property connects a ComicContentElement to a comic.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasLayer' '"
<rdfs:domain rdf:resource="#Panel' '/>
<rdfs:range rdf:resource="#Layer' '/>
<rdfs:comment>This property makes a layer part of a panel.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="belongsToPanel' '"
<owl:inverseOf rdf:resource="#hasLayer' '/>
<rdfs:comment>This property tells to which panel a layer belongs.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasPanel' '"
<rdfs:domain rdf:resource="#Page' '/>
<rdfs:range rdf:resource="#Panel' '/>
<rdfs:comment>This property makes a panel part of a page.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasPage' '"
<rdfs:domain rdf:resource="#Comic' '/>
<rdfs:range rdf:resource="#Page' '/>
<rdfs:comment>This property makes a page part of a comic.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="belongsToPage' '"
<owl:inverseOf rdf:resource="#hasPanel' '/>
<rdfs:comment>This property tells to which page a panel belongs.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasLocation' '>

99
This property tells if the scene of a certain panel is located on a certain location.

This property tells to which layer uses a certain location.

This property tells if a layer represents a certain character, 1 layer can represent multiple characters.

This property tells to which layer uses a certain character.

This property tells if a layer represents a certain object, 1 layer can represent multiple objects.

This property tells to which layer uses a certain object.

This property tells if a layer represents audio, 1 layer can represent multiple audio objects.
<owl:ObjectProperty rdf:ID="audioUsedOn">
    <owl:inverseOf rdf:resource="#hasAudio"/>
    <rdfs:comment>This property tells to which layer uses a certain audio.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasTextElement">
    <rdfs:domain rdf:resource="#Layer"/>
    <rdfs:range rdf:resource="#ComicTextElement"/>
    <rdfs:comment>This property tells if a layer represents a certain ComicTextElement, 1 layer can represent multiple ComicTextElement.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="textElementUsedOn">
    <owl:inverseOf rdf:resource="#hasTextElement"/>
    <rdfs:comment>This property tells to which layer uses a certain TextElement.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasText">
    <rdfs:domain rdf:resource="#Layer"/>
    <rdfs:range rdf:resource="#Text"/>
    <rdfs:comment>This property tells if a layer displays text.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="textUsedOn">
    <owl:inverseOf rdf:resource="#hasText"/>
    <rdfs:comment>This property tells to which layer uses a certain Text.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="says">
    <rdfs:domain rdf:resource="#Text"/>
    <rdfs:comment>This property denotes the actual text that is being said by a Text class.</rdfs:comment>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="isBalloonFrom">
    <rdfs:domain rdf:resource="#Balloon"/>
    <rdfs:range rdf:resource="#Character"/>
    <rdfs:comment>This property tells to which character a certain speech balloon belongs.</rdfs:comment>
</owl:ObjectProperty>
A.3 SPARQL Queries

In this section we give a number of examples of SPARQL queries that can be used to fetch information from a comic. The query in Listing A.3 fetches the names of all the characters in a comic. To fetch all the panels together with their location, the query in Listing A.4 can be used. The names of the locations are presented in Dutch, however, this can also be done in English if ‘nl’ is replaced by ‘en’ on line 10.
Listing A.3: SPARQL query to fetch the names of all characters

```sparql
PREFIX dicera: <http://semweb.mmlab.be/ns/dicera#>

SELECT *
WHERE {
  ?test a dicera:Character .
  ?test dicera:hasName ?x .
}
```

Listing A.4: SPARQL query to fetch all the panels together with their location (in Dutch)

```sparql
PREFIX dicera: <http://semweb.mmlab.be/ns/dicera#>

SELECT ?panel (str(?name) as ?location)
WHERE {
  ?panel a dicera:Panel .
  ?panel dicera:hasLayer ?layer .
  ?layer a dicera:Layer .
  ?layer dicera:hasLocation ?loc .
  ?loc a dicera:Location .
  ?loc dicera:hasName ?name .
  FILTER(langMatches(lang(?name), 'nl'))
}
```
Appendix B

Layering through Canvas

HTML5 has an element called ‘canvas’\(^1\). Multiple \(<\text{canvas}>\) elements can be layered upon each other\(^2\). The order of the different layers is defined by the z-index CSS property. The negative side and the reason we abandoned this approach is the burden that is being put on the JavaScript. More code has to be run to only display the different images: for every layer his context is needed and the image is drawn on the canvas. Using the CSS approach described in Section 4.2, we simplify the use of layering and remove the need for suchlike coding.

\(^{1}\text{http://html5.litten.com/simple-animation-in-the-html5-canvas-element/}\)
\(^{2}\text{http://html5.litten.com/using-multiple-html5-canvases-as-layers/}\)
Appendix C

Frameworks and Libraries for Use Cases and Desktop Application

The sections in this appendix provide more information about the frameworks and libraries used in the Java applications of Section 6.2. Their use can be found on the accompanied CD in the folder code.

C.1 Restlet

To support the communication between the websites and the Java application we use the Restlet\(^1\) framework. It enables us to easily create a REST API, upon the Java application, that can be contacted by the websites.

C.2 Jena

To extract and reason about the information embed in the EPUB file, we use the Apache Jena\(^2\) framework. This framework allows us to create RDF graphs and allows to read the information from those graphs to answer the different queries sent by the websites and issued by the desktop application. Because we use RDFa to present our information, we need Semargl\(^3\) which allows Jena to read the information in the XHTML pages of the EPUB file, so it can generate the graphs. Once we have the graph, we can use the Jena model to retrieve the necessary information or we can create a SPARQL query. Both options can answer the different queries, however, the SPARQL option allows for a more

\(^1\)http://restlet.org
\(^2\)https://jena.apache.org
\(^3\)http://semarglproject.org
framework independent implementation: the use of the Jena model is limited to the Jena framework and the SPARQL queries can be used outside of the Jena framework.

C.3 jsoup

For extracting the images used in the layers (of the panels), we use jsoup\(^4\). This Java library allows use to easily select the correct \(<\text{div}\>\) element containing the \(<\text{img}\>\) element of the required layer.

\(^4\)http://jsoup.org
Appendix D

List of EPUB Books to Determine Average Size

- Alexander, Eben - Proof Of Heaven
- Albom, Mitch - The First Phone Call From Heaven
- Andrews, V.C. - Flowers In The Attic
- Aspe, Pieter - 13
- Aspe, Pieter - Alibi
- Aspe, Pieter - Bankroet
- Aspe, Pieter - Blauw Bloed
- Aspe, Pieter - Casino
- Aspe, Pieter - De Cel
- Aspe, Pieter - De Japanse Tuin
- Aspe, Pieter - De Kinderen Van Chronos
- Aspe, Pieter - De Midasmoorden
- Aspe, Pieter - De Vierde Gestalte
- Aspe, Pieter - De Vijfde Macht
- Aspe, Pieter - De Zevende Kamer
- Aspe, Pieter - Dood Tij
- Aspe, Pieter - Grof Wild
- Aspe, Pieter - Het Dreyse-inicident
- Aspe, Pieter - Het Vierkant Van De Wraak
- Aspe, Pieter - Misleid
- Aspe, Pieter - Onder Valse Vlag
- Aspe, Pieter - Ontmaskerd
- Aspe, Pieter - Onvoltooid Verleden
- Aspe, Pieter - Op Drift
- Aspe, Pieter - Pandora
- Aspe, Pieter - Rebus
- Aspe, Pieter - Tango
- Aspe, Pieter - Vagevuur
- Aspe, Pieter - Zoenoffer
- Aspe, Pieter - Zonder Spijt
- Baantjer - De Cock en ’t Wassend Kwaad
- Baantjer - De Cock en Danse Macabre
- Baantjer - De Cock en De blijde Bacchus
- Baantjer - De Cock en De Bloedwraak
- Baantjer - De Cock en De Broeders Van De Haat
- Baantjer - De Cock en De Broeders Van De Zachte Dood
- Baantjer - De Cock en De Dartele Weduwe
- Baantjer - De Cock en De Dode Harlekijn
- Baantjer - De Cock en De Dode Tempeliers
- Baantjer - De Cock en De Dood In Antiek
- Baantjer - De Cock en De Dood In Gebed
- Baantjer - De Cock en De Dood Van de Heelende Meesters
- Baantjer - De Cock en De Dood Van Een Clown
- Baantjer - De Cock en De Dood Van Een Kunstenaar
- Baantjer - De Cock en De Dood Van Een Profeet
- Baantjer - De Cock en De Dwaze Maagden
- Baantjer - De Cock en De Gebrandmerkte Doden
- Baantjer - De Cock en De Moord In Brons
- Baantjer - De Cock en De Moord In Extase
- Baantjer - De Cock en De Moord In Seance
- Baantjer - De Cock en De Moord In Triplo
- Baantjer - De Cock en De Moord Op Anna
Bentveld
• Baantjer - De Cock en De Naakte Juffer
• Baantjer - De Cock en De Ondergelijke Dood
• Baantjer - De Cock en De Onthoofdelijke Dode
• Baantjer - De Cock en De Romance In Moord
• Baantjer - De Cock en De Sluimerende Dood
• Baantjer - De Cock en De Smekende Dood
• Baantjer - De Cock en De Treurende Kater
• Baantjer - De Cock en De Wortel Van Het Kwaad
• Baantjer - De Cock en De Wurger Op Wondag
• Baantjer - De Cock en De Zorgvuldige Moordenaar
• Baantjer - De Cock en Dood Door Hamerslag
• Baantjer - De Cock en Een Deal Met De Duivel
• Baantjer - De Cock en Een Dodelijk Rendez-vous
• Baantjer - De Cock en Een Dodelijke Dreiging
• Baantjer - De Cock en Een Duivels Komplot
• Baantjer - De Cock en Een Recept Voor Moord
• Baantjer - De Cock en Een Strop Voor Bobby
• Baantjer - De Cock en Een Veld Papavers
• Baantjer - De Cock en Geen Excuses Voor Moord
• Baantjer - De Cock en Het Dodelijk Akkoord
• Baantjer - De Cock en Het Duel In De Nacht
• Baantjer - De Cock en Het Lijk Aan De Kerkmuur
• Baantjer - De Cock en Het Lijk In De Kerknacht
• Baantjer - De Cock en Het Lijk Op Drift
• Baantjer - De Cock en Het Lijk Op Retour
• Baantjer - De Cock en Het Masker Van De Dood
• Baantjer - De Cock en Het Roodzijden Nachthemd
• Baantjer - De Cock en Het Sombere Naakt
• Baantjer - De Cock en Kogels Voor Een Bruid
• Baantjer - De Cock en Moord A’la Carte
• Baantjer - De Cock en Moord Eerste Klasse
• Baantjer - De Cock en Moord In De Hondsdagen
• Baantjer - De Cock en Moord In Reclame
• Baantjer - De Cock en Moord Op Bestelling
• Baantjer - De Cock en Moord Op De Bloedberg
• Baantjer - De Cock en Moord Op Termijn
• Baantjer - De Cock en Tranen Aan De Leie
• Baantjer - Rechercheur Versteeg En De Derden Katten
• Baantjer & De Waal - Een Rus In De Joradaan
• Baantjer & De Waal - Een Dief In De Nacht
• Baantjer - De Cock en De Dansende Dood
• Baantjer - De Cock en De Ganzen Van De Dood
• Baantjer - De Cock en De Stervende Wandelaar
• Baantjer - De Cock en Moord In Beeld
• Baker Kline, Christina - Orphan Train
• Baldacci, David - King And Maxwell
• Belfort, Jordan - The Wolf Of Wall Street
• Bos, Geertje - Baantjes alias De Cock
• Black, Campbell - Indiana Jones En De Verloren Ark
• Blair, Annette and others - Scandalous Brides
• Bradley, Alan - The Dead In Their Vaulted Arches
• Burpo, Todd with Vincent, Lynn - Heaven Is For Real
• Cain, Susan - Quiet
• Charmed - De Macht Van Drie
• Charmed - Tuin Van Het Kwaad
• Charmed - Rendez-Vous Met De Dood
• Charmed - Duistere Wraak
• Christie, Agatha - Moord Op Golflinks
• Christie, Agatha - Moord Uit Het Verleden
• Clancy, Tom with Greaney, Mark - Command Authority
• Connelly, Michael - The Gods Of Guilt
• Connelly, Michael - Switchblade
• Dahl, Roald - Boy 1916-1937
• Dahl, Roald - Het Wonderlijke Verhaal Van Hendrik Meier
• Dahl, Roald - Matilda
• Dahl, Roald - Oom Oswald
Bibliography


115


List of Figures

1.1 Comic book market revenue ............................................. 2

3.1 Solution overview ....................................................... 19

4.1 Animation sequence .................................................... 24

6.1 Character-Driven Comic Search Engine with Geographical Visualization - Search form .................................................. 53

6.2 Character-Driven Comic Search Engine with Geographical Visualization - Results shown as a grid ..................................... 54

6.3 Character-Driven Comic Search Engine with Geographical Visualization - Results shown on a map ..................................... 54

6.4 Character-Driven Panel Extractor for Comic - List of comics ....... 55

6.5 Character-Driven Panel Extractor for Comic - Selected comic ‘Het Sprekende Testament’ with list of featured characters ................. 56

6.6 Character-Driven Panel Extractor for Comic - Panels of selected character ‘Jerom’ ....................................................... 56

6.7 ComicIV - Basic Information ............................................ 57

6.8 ComicIV - Characters ..................................................... 57

6.9 ComicIV - Locations ....................................................... 58

6.10 ComicIV - Genres ......................................................... 58

6.11 ComicIV - SPARQL ....................................................... 59

6.12 ComicIV - Text Search ................................................... 59

7.1 Single layer panel .......................................................... 62

7.2 Result of dissecting a character from a single layer panel .......... 63

8.1 Graph comparing file size different types of comics ................. 72
8.2 Graph comparing the file size of custom comic and ‘Suske en Wiske’ comic 72
8.3 Proposed solution on Kobo Glo, using fallback panels . . . . . . . . . . . . 75