Frederik R. N. Schlupkothen

A Genre-aware Document Model for Multichannel Publishing Workflows

Frederik R. N. Schlupkothen

A Genre-aware Document Model for Multichannel Publishing Workflows

Dissertation

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

vorgelegt an der Fakultät für
Elektrotechnik, Informationstechnik und Medientechnik

an der
Bergischen Universität Wuppertal

Mai 2016

Die Dissertation kann wie folgt zitiert werden:

urn:nbn:de:hbz:468-20161019-121805-0
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20161019-121805-0]

An dieser Stelle möchte ich all denjenigen danken, die mich in Ausbildung und Beruf mit Rat und Rückhalt unterstützt und geprägt haben – die vorliegende Arbeit ist auch das Ergebnis ihrer Mühen. Im direkten Kontext dieser Arbeit möchte ich besonders danken:

Für die engagierte Betreuung meiner Arbeit und die stets gut gelaunte Zusammenarbeit im Schoß der Druck- und Medientechnologie, Prof. Dr. *Karl-Heinrich Schmidt* (BU Wuppertal).

Für die Bereitstellung des Gannet-Beispiels und die Möglichkeit zum persönlichen Austausch, Prof. Dr. *John A. Bateman* (Universität Bremen).

Für die Möglichkeit zum persönlichen Austausch „auf halber Strecke" in Gießen, Prof. Dr. *Marko Hedler* (HdM Stuttgart).

Für die Einführung in die frankophone Dokumententheorie und deren Publikmachung im Deutschsprachigen, Prof. Dr. *Stefan Gradmann* (KU Leuven).

For transmitting a playful yet serious approach to information spaces of many kinds and for being great cosmopolitan fellows, *Michael Cohen* (Prof., Ph. D. / The University of Aizu) and *Yamadera Jun* (Eyes, JAPAN Co. Ltd.).

Für die Ermutigung und die Unterstützung beim Angehen des Promotionsvorhabens, Prof. Dr. *Alexander Stoffel* (TH Köln) und Prof. Dr. *Carsten Vogt* (TH Köln).

Für die Einführung in Forschung und Entwicklung in Bezug elektronischer Dokumentenverarbeitung, einer Thematik der ich in meinem weiteren Werdegang treu geblieben bin, *Heike Horstmann*, *Thomas Tikwinski* und dem Kollegium am Fraunhofer IMK / IAIS.

Für die gute Atmosphäre und den Zusammenhalt beim Bewältigen gemeinsamer Aufgaben und Herausforderungen, meinem Kollegen *Gilles Bülow*.

Für den privaten Rückhalt, meinen Eltern, meinen Geschwistern.

– Vielen herzlichen Dank!

Abstract

Based on the experience that the classic separation between logical and layout view on document contents is not enough to realize general single source and cross-media publishing solutions, this thesis investigates a document model to conceptualize a new multichannel publishing workflow.

At first, the thesis introduces theoretical frameworks of analyzing and defining documents that potentially take different document types into account. Then, these informal frameworks are discussed and combined to define a formal, genre-aware document model. Next, the resulting model is implemented in a prototypical document processing environment that complies with common standards. Finally, the functionality of the new document processing workflow is tested by defining specific genres and creating corresponding documents.

Zusammenfassung

Die Erfahrung hat gezeigt, dass die klassische Trennung zwischen logischer Sicht und Layoutsicht auf Dokumente im Allgemeinen nicht reicht, um unterschiedliche medienübergreifende Publikationen aus einem medienneutralen Quellformat zu erzeugen. Daher entwirft die vorliegende Arbeit ein neues Dokumentenmodell zur Konzeptualisierung allgemeiner mehrkanaliger Publikationssysteme.

Zunächst werden theoretische Beschreibungsmodelle zur Dokumentenanalyse und -definition vorgestellt, die unterschiedliche Dokumententypen in Betracht ziehen. Diese informellen Modelle werden diskutiert und zu einem formalen „Genre-abhängigen“ Dokumentenmodell zusammengeführt. Der so beschriebene Dokumentenbegriff wird anschließend prototypisch auf Grundlage etablierter Publikationsstandards und -werkzeugen umgesetzt. Abschließend wird die Funktionalität des neuen Publikationssystems anhand konkreter Fallbeispiele gezeigt.

Résumé

L'expérience a montré que la distinction classique entre le regard porté sur la logique et celui porté sur la mise en page des documents ne suffit pas généralement à produire des publications pour différents médias en partant d'un format d'origine neutre médialement parlant. C'est pourquoi la thèse suivante conçoit un nouveau modèle de document pour la conceptualisation de systèmes de publication multiple pour différents médias.

D'abord, nous présentons des modèles de description théoriques pour analyser ou définir des documents de types différents. Ensuite, ces modèles informels sont discutés et nous les amenons à une représentation formelle de document qui relève du genre. Cette notion de document décrite ainsi est ensuite mise en application sur la base de standards et de systèmes de publication établis. Enfin, nous montrons la fonctionnalité du nouveau système de publication à l'aide d'exemples concrets.

Contents

List of Figures

List of Tables

List of Definitions

List of Listings

Acronyms

AR Augmented Reality. 111

CJK Chinese, Japanese, and Korean. 24, 25, 55, 73

DAG Directed Acyclic Graph. 18, 20, 29, 63, 74, 92, 215

DOM Document Object Model. xvi

DTD Document Type Definition. 47, 48, 62, 79, 80, 105, 113, 124, 133, 224

HTML HyperText Markup Language. xvii, 57–59, 104–106, 108, 110, 113, 114

IRI Internationalized Resource Identifier. 46, 48, 51, 53, 56, 75

NCX Navigation Control file for XML applications. 76, 113

PAULA Potsdamer Austauschformat Linguistischer Annotationen. 44

PDF Portable Document Format. 82, 102, 104

RDF Resource Description Framework. xvi, 44–46, 48–50, 52, 53, 63, 92, 112

RST Rhetorical Structure Theory. 14, 16, 76–81, 83–88, 90, 91, 93, 94, 97–99, 101, 102, 104, 108, 113, 124, 134, 174, 177, 208, 224–226, 231–233

SVG Scalable Vector Graphics. 79, 108, 233

TEI Text Encoding Initiative. 3, 44, 114

TTS Text to Speech. 106

Turtle Terse RDF Triple Language. 46

UML Unified Modeling Language. 61, 74, 90

VR Virtual Reality. 110

W3C World Wide Web Consortium. 4, 42

WWW World Wide Web. xvi, 10, 11, 45

X3D Extensible 3D. xvi, 108, 110

X3DOM X3D Document Object Model. 109

XCES Corpus Encoding Standard for XML. 44

XHTML Extensible HTML. 48, 54–61, 76, 83, 84, 87, 89, 96, 97, 102–105, 107, 113, 114, 124, 133

XInclude XML Inclusions. 44

XLink XML Linking Language. 44–55, 57, 58, 60, 61, 63–67, 71, 74, 75, 83, 84, 86, 87, 92–94, 96, 97, 99, 112, 124, 133

XML Extensible Markup Language. xvi, xvii, 1, 4, 9, 18, 30, 42–47, 51–53, 55, 61–63, 65–71, 73–76, 78–80, 82–84, 92, 93, 97, 106, 108, 112, 113, 124, 131, 132, 144

XPath XML Path Language. 74, 78, 79, 82, 83, 90–92, 100, 101

XPointer XML Pointer Language. 44, 75

XProc XML Pipeline Language. 64, 74, 80, 89, 90, 92, 102, 104, 108, 113

XSD XML Schema Definition. 63

XSL Extensible Stylesheet Language. xvii, 1, 78, 91

XSLT XSL Transformation. 40, 51–53, 64, 66, 67, 70, 71, 74–76, 78–83, 87–92, 100, 102, 104, 108, 113

1. Introduction

A core principle of document management within the publishing industry is the media-independent storage of contents. In the common understanding of this principle, a document is not managed upon its specific output form, but upon a structure that organizes the content according to its reading-logic, leaving the flexibility to apply alternative layout rules. The methodical utilization of workflows that implement this principle to create alternative publications of different media types from a single content base is called "single-source", "cross-media", or "multi-channel" publishing. (See e.g. Sandkuhl 1996, Möhr and Schmidt 1999, Ott 2013)

The Extensible Markup Language (XML) has become the native markup language for structured information in (distributed) document processing architectures. It provides a core syntax that has been adopted by many document description languages. A broad software ecosystem and further standards have emerged to realize and facilitate the processing of XML-based documents. Among them, the Extensible Stylesheet Language (XSL) Family offers a standardized mechanism to translate different XML-based languages into each other. This has made XML the language of choice for cross-media publishing workflows. (See Bray et al. 2006, Berglund 2006)

The growing variety of reading devices (e.g. smart phones, tablets, e-book readers) creates new possible publication channels, but leads to a diversification of publication types as well. Accordingly, automated multichannel publishing workflows can assist the generation of new publication products out of preexisting documents. As described before, these workflows are based on the separation of content and layout. But the transfer of content to different media is not only a presentation shift. Different output contexts come with different expectations and conventions for document usage. As an example, a book and a corresponding web page are expected to work differently in terms of reading and navigation experience. But the reading-logic and the navigation-structure are usually implicitly predefined in the logical organization

of the document. Furthermore, the content modalities (i.e. the content realization as e.g. text, image, or audio) are usually predetermined by the document description. This lack of generalization in content representation restricts the automated generation into a specific type of document. That is, if a source document description relies on a specific content modality, all derived output documents are restricted to use the same original content realizations or preconfigured alternatives. A typical example is an image in a web page and a preconfigured description text—the transformation of the image's content to video is not possible in actual multichannel publishing systems.

The notion of document has a history of redefining what actually constitutes a document. Recently alternative approaches are widely discussed. In order to enhance the automation possibilities of multichannel publishing solutions an investigation of alternative document models is necessary. Therefore, this thesis investigates an alternative document model to conceptualize and implement a general multichannel publishing workflow by considering two questions:

1. How to constitute a document model such that the content organization is strictly separated from the content representation?
2. How to implement a new document model such that it is deployable within standardized publishing environments?

In this thesis these questions are addressed as described in the following section.

1.1. Research Objective

To start with an example, consider a scientific article, its corresponding poster presentation and slide presentation. Even though all three documents can communicate the very same information in their presentation environment, the actual documents are very different from each other in terms of their form. The article and its flow is typically text-based, supported by graphical elements, and paginated; the poster is typically graphically organized and supported by reduced text; the presentation and its flow is speech-based and supported by paginated text and graphical elements—while it is intuitively clear to the author and the audience that in terms of communicated information all three documents rely on a common foundation, it is difficult to describe this relation in the actual document models

used in publication workflows. What can be said is that the three derivatives "scientific article", "poster" and "slide set" are "hierarchically structured representations of [. . .] content objects" that "survive[] not only changes of layout, printing technology, and so on, but even translation" (DeRose et al. 1990, p. 6 f.)[1] and that these representations belong to different subgenres, where "exemplars of a genre exhibit various patterns of similarity in terms of structure, style, content and intended audience" (Swales 1990, p. 58).[2]

In many text based approaches to documents (e.g. the Text Encoding Initiative (TEI), see Burnard and Bauman (2015)) the concept of a genre is decisive for the work on genre-sensitive document languages. Especially, similarities and differences of document types are investigated under the notion of genre and well-established. The transfer of these established concepts of genre to document generation and publication workflows for non-textual documents should give insight on how multimedia publishing solutions could be enhanced or how automated document generation could be advanced.

Therefore, this thesis investigates what constitutes a genre and how it can be defined and modeled generally in terms of structured documents. Based on theoretical assumptions, a prototypical framework is implemented that can describe and define different document-genres and realize corresponding documents that comply to these genres. Finally, experiments that include explicitly non-textual document representations exemplify how a specific document of a specific genre can be transferred from one derivative subgenre to another. Generally speaking, this new approach leads to the question whether the traditional focus on media formats can be replaced with the focus on genre in order to conceptualize (automated) multichannel publishing solutions.

Thus, based on the experience that the classic separation between logical view and layout view on document contents is not enough to realize general single source and cross-media publishing solutions, this thesis investigates an alternative foundation to describe documents by the following procedure.

1. While the original source refers to the translation between different spoken languages, here we think beyond that of the translation between different modalities.
2. For a full description of genre see Appendix A.1. Note, that Swales (2004, p. 61) is questioning this description as being too restrictive and prefers to allow alternative (complementary) characterizations of genre. However, in the context of document modeling a deliberate simplification of the concept is desirable.

1.2. Outline

After this introduction (Chapter 1), the following Chapter 2 investigates the theoretical foundations by exploration of different approaches to analyze, describe, and compare documents. Extensive studies in the recent past that take the technical perspective of document descriptions into account have been formulated e.g. by Bateman (2008) and the researcher collective Pédauque (2006a, 2007).

Being rather analytical frameworks, these models form the basis in the creation of a new model to generate documents. The main point of the subsequent theoretical work is how generic document types (i.e. genres) can be defined in relation to each other and in what they differ. In this process the selection criterion is driven by the model's usability in publishing environments. Therefore, the informal frameworks discussed in Chapter 2 are combined to define a formal, genre-aware document model in Chapter 3.

In Chapter 4, the resulting model is re-formulated by means of computable formats and technologies as proposed by the World Wide Web Consortium (W3C), i.e. modeled according to XML-based standards. Furthermore, not only the modeling, but a working prototypical environment to process corresponding documents is conceptualized on the basis of freely available software frameworks.

Chapter 5 demonstrates the functionality of the proposed document model and the resulting prototype in the context of multichannel publishing by defining specific genres and creating corresponding documents.

Finally, Chapter 6 concludes the thesis and provides an outlook to possible follow-up studies.

The secret to productivity in so many fields is letting dead people do your work for you.
— Robert J. Lang

2. Towards a Genre-aware Document Model[1]

The concept of document as "recorded information or material object which can be treated as a unit in a documentation process" (ISO 5127 2001, p. 11) plays a central role in contemporary publishing. But the notion of document has a history of redefining what actually constitutes a document and is discussed by different scientific disciplines that offer a broad field of study. Originating from the Latin word "documentum" a document could be anything "serving the act of instruction, teaching, and giving a lecture" (Lund and Skare 2009, p. 1632). Over the centuries the notion of document is more and more related to a physical matter holding written evidence and serving as empirical proof. In the end of the 19th century the first specific document theories are conceptualized in the context of libraries and mark the beginning of a scientific approach towards the notion of document. As the creation and distribution of documents is a basic cultural technique of human interaction and communication, the concept of document has been subject to analysis in many scientific disciplines. Three major approaches are briefly summarized below in Section 2.1. Next, Sections 2.2–2.4 introduce recent theories of analyzing and defining documents that agree with three complementary views on documents: the perceptual, meaning-related, and communicative aspects of a document. These theoretical frameworks form the basis of a new document model that is concluded in Section 2.5 and that will be used to conceptualize a multichannel publishing workflow in the following chapters.

2.1. Major Approaches to Documents

Significant early examples of document concepts are provided in the context of libraries by Paul Otlet (1868–1944) and Suzanne Briet (1894–1989). Otlet (1934) defines a framework of documentation sketched in Figure 2.1 on the next page that describes documents as

1. Parts of this chapter are based on Schlupkothen and Schmidt (2014).

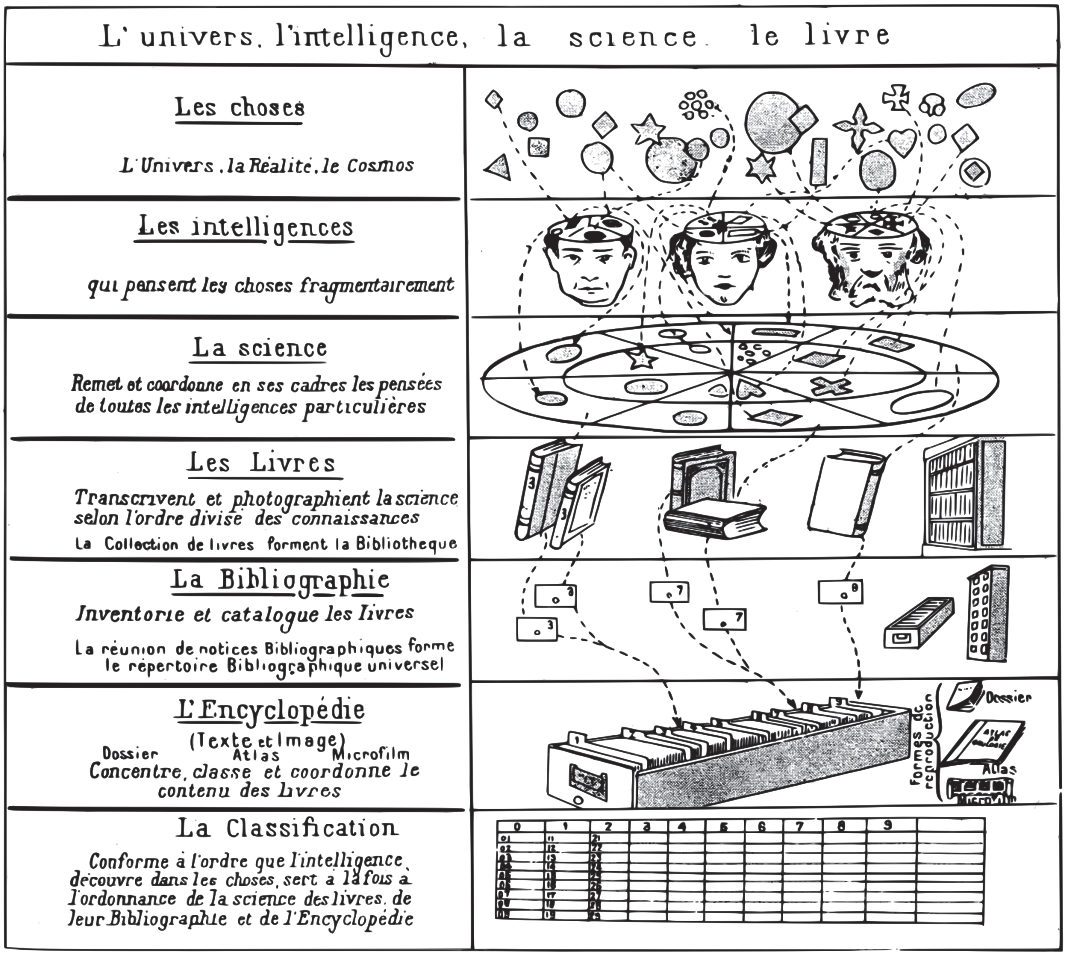


Figure 2.1.: *Framework of documentation as described by Paul Otlet (Otlet 1934, p. 41)*

shared knowledge and thoughts about a perceived reality. The content of these documents is implicitly related to each other by means of fact-cards that describe and categorize each document's content. Otlet's idea was to make these relations explicit resulting in an universal encyclopedia that describes all known facts. The main focus of this framework is on graphic-centered documents (especially printed matters), but is explicitly open to other forms of documentation (e.g. sound recordings or natural objects). With Briet (1951) the definition of a document is drawn back from a form- to a function-related description. She defines a document as "any concrete or symbolic indexical sign, preserved or recorded toward the ends of representing, of reconstructing, or of proving a physical or intellectual phenomenon" (Briet 1951, p. 7 as translated in Day et al. 2006, p. 10). Briet's famous example of a document is an antelope: not an antelope in the wild, but a caught and studied one that is shown in a zoo, being a physical proof and documentation of itself. Scholar articles about that

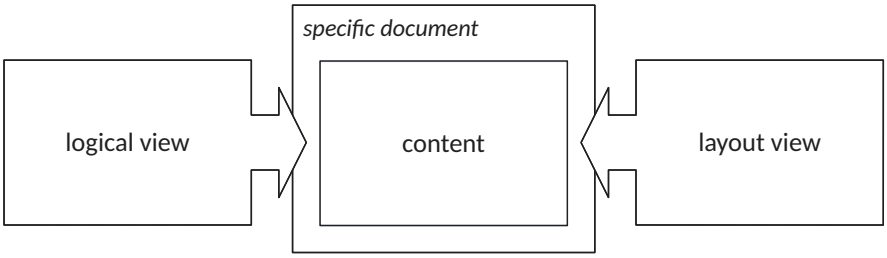


Figure 2.2.: *Views of a document (cf. ISO/IEC 8613-2 1993, Fig. 1)*

antelope are considered secondary documents, derived from the antelope being the initial one. Following Otlet's and Briet's approaches the scientific field of document theory emerged and keeps discussing what documents are, what they are about, and what they do. (See particularly Buckland 1997, Lund and Skare 2009)

A second scientific field that focuses on the analysis of documents is given by different disciplines of the humanities. Especially two disciplines are significant in the context of documents: linguistics and the psychology of perception. Linguistics focus on the content of text documents and define analytical frameworks around the notion of text "with the aim of building up models of types of texts and statements (narrative, description, argument, instruction, etc.)" (Virbel 1989, p. 164). Text as a written statement or discourse is segmented into signs that are in relations to each other, to their meaning, and to their users. These relations can be described by the notions of syntax, semantics, and pragmatics that can be applied not only to textual entities but to document structures that comprise discourse in general. The psychology of perception focuses on the perceptual aspects of documents. Important questions are how documents are decomposed by a user and therefore how document design supports the understanding of document content. For example the so-called gestalt theory formulates rules of pattern recognition in visual perception and how visual entity grouping is performed by the human visual system. These disciplines have led to inter-disciplinary studies on multimodality that aim to identify and understand distinct "modes" of communication (written, spoken, gestural, etc.) and how these modes are used to convey meaning. (See particularly Bateman 2008)

A third scientific field covers the disciplines of computer and information science that design and realize systems of (automated) document generation and processing. Commonly, these systems describe a document by means of a hierarchical organization that represents a specific view of the content. Figure 2.2 above sketches the typical approach in the process

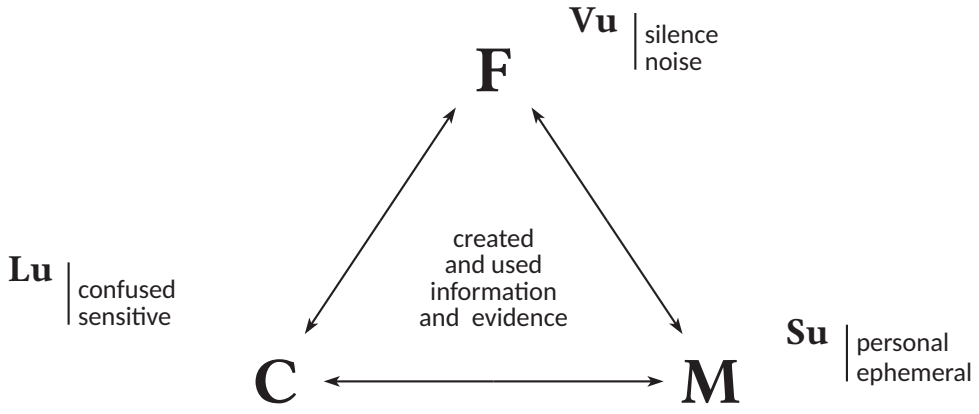


Figure 2.3.: *"Vu, lu, su"-model of a document based on the notions of form (F), content (C), and medium (M), (cf. Pédauque 2006a, p. 20, translated by F. S.)*

of document creation that separates a logical view from the layout view. Accordingly, the content is organized in a revisable format that follows the document's reading logic; this leaves the flexibility to apply alternative sets of output rules to transform the content into different output formats with different layout (e.g. paginated or scrollable, static or dynamic representations). To conceptualize automated document processing systems, either document organization can be restricted by means of modeling languages that describe the expectable generic structures of a document that belongs to a specific type (e.g. books or web sites). Due to the growing variety of output channels (e.g. smart phones, tablets, e-book readers) these document types are more and more diversified. Therefore, strategies to generate different publication products out of a single source are conceptualized in terms of multichannel publishing workflows. (See particularly André et al. 1989, Ott 2013)

This leads us to recent models of analyzing documents.

2.2. Form, Content, Medium

An interdisciplinary francophone researcher collective under the pseudonym of Roger T. Pédauque has taken the phenomenon of digitization into account to re-define the notion of document (Pédauque 2006a, 2007). Derived from the linguistic approach to describe signs of a language under the three different relations of (morpho-)syntax, semantics, and pragmatics, they outline three dimensions that define a document. Figure 2.3 above

summarizes the document model in views of the form or sign that is opposed to the silence or noise, the text or content that is opposed to the confused or sensitive, and the medium or relation that is opposed to the personal or ephemeral. In other words: an information has to be perceptible, understandable, and communicable in order to be a document. These dimensions are described in detail below. (See particularly Pédauque 2006b)

The *form* dimension focuses on the anthropological aspect of a document, i.e. how a user as a physical being with senses interacts with the document as an object. Thus, a reader has to perceive and understand a deployed physical and logical structure (e.g. pages, chapters, indexes) in order to be able to use a document. Pédauque describes a form-related change from the object-centered, traditional notion of documents to their digitized counterparts. This notion shift is described by three definitions: First, a traditional document is defined by its carrier and inscription. This unity is obvious for printed objects, as the physical and logical structures are consistent with each other. Yet audiovisual documents typically need a player to reconstruct the document and to make it perceptible. However, this matter is even more complicated for electronic documents, where the processing systems influence the document's structure (i.e. the description language, bits and bytes) and different output from a single source is possible. Therefore, an electronic document is defined by its structure and data—or more specifically: "An electronic document is a data set organized in a stable structure associated with formatting rules to allow it to be read both by its designer and its readers" (Pédauque 2006b, p. 45 as translated in Pédauque 2003, p. 9). And finally, an XML document is defined by its structured data and separated layout. These definitions lead the attention to the legibility and the perception of a document. This aspect of the document as an object that can be held and perceived is emphasized in the name of this dimension: "vu", the French word for "the seen".

Hence, the "vu"-dimension of a "document as form" can be summarized as follows:

traditional document = carrier + inscription[2]

electronic document = structures + data

XML document = structured data + formatting

From an economical perspective, the form dimension is the main value-added model of publishers, as they rely on selling goods. Thus, publishers assist the creation of copyrighted

2. Originally translated with "*medium* + inscription" (Pédauque 2003). Here changed to "*carrier* + inscription" in conformance with the original french notion "support" (Pédauque 2006b) to clearly distinguish the "vu" dimension (*form*/sign) from the later described "su" dimension (*medium*/relation).

works in order to monetize reproductions. In the context of electronic documents and the World Wide Web (WWW), companies like Apple that focus on selling reading devices rely on the form dimension. (See Salaün and Collegium de Lyon 2011, Salaün 2012, passim)

The *content* dimension focuses on the intellectual aspect of a document, i.e. how a user as a cognitive being understands the document as a meaning carrier. Thus, a reader has to decipher and construe the information from a given coding system (e.g. written text) with the help of contextual knowledge (e.g. domain-specific knowledge) in order to be able to understand a document. Pédauque describes a content-related change from the traditional notion of documents to their digitized counterparts. This notion shift is described by the following definitions: First, a traditional document is defined by its inscription and meaning, where all interpretation, contextualization, and reconstruction of the embedded meaning is left to the reader. With the digitization a document becomes accessible to mining techniques that realize automated knowledge extraction (e.g. by means of indexing or classification). Therefore, an electronic document is defined by its informed text and the derived knowledge to a user—or more specifically: "An electronic document is a text whose elements can potentially be analyzed by a knowledge system in view of its exploitation by a competent reader" (Pédauque 2006b, p. 59 as translated in Pédauque 2003, p. 15). And finally, a semantic web document is defined by its informed text and (underlying) ontology. These definitions lead the attention to the understanding and the assimilation of a document. This aspect of the document as information that can be understood and interpreted is emphasized in the name of this dimension: "lu", the French word for "the read".

Hence, the "lu"-dimension of a "document as content[3]" can be summarized as follows:

$$\begin{aligned} \text{document} &= \text{inscription} + \text{meaning} \\ \text{electronic document} &= \text{informed text} + \text{knowledge} \\ \text{semantic web document} &= \text{informed text} + \text{ontologies} \end{aligned}$$

From an economical perspective, the content dimension is the main value-added model of libraries, as they rely on providing access to knowledge. Thus, libraries enable group members to achieve a joint objective. In the context of electronic documents and the WWW, companies like Google that focus on the exploitation of information rely on the content dimension. (See Salaün and Collegium de Lyon 2011, Salaün 2012, passim)

3. Originally named "document as *sign*" to emphasize on the document as a "meaningful object" (Pédauque 2003, 2006b). Here changed to "document as *content*" to clearly distinguish the "lu" dimension (text/*content*) from the previously described "vu" dimension (form/*sign*).

The *medium* dimension focuses on the social aspect of a document, i.e. how a user as a member of a society shares and values a document. Thus, a reader has to trust a document (e.g. as an authority: invoice, quotation, etc.) as it is transferred in space and time in order to use a document. Pédauque describes a medium-related change from the traditional notion of documents to their digitized counterparts. This notion shift is described by the following definitions: First, a traditional document is defined by its inscription and legitimacy, as a document gives status to an information by making it accessible and persistent. An electronic document is defined by its text and procedure—or more specifically: "An electronic document is a trace of social relations reconstructed by computer systems" (Pédauque 2006b, p. 74 as translated in Pédauque 2003, p. 23). And finally, a web document is defined by its publication and observed access. These definitions lead the attention to the sociability and the social embedding of a document. This aspect of the document as information that can be transferred in time and space is emphasized in the name of this dimension: "su", the French word for "the known".

Hence, the "su"-dimension of a "document as medium" can be summarized as follows:

$$\begin{aligned} \text{document} &= \text{inscription} + \text{legitimacy} \\ \text{electronic document} &= \text{text} + \text{procedure} \\ \text{web document} &= \text{publication} + \text{cued access} \end{aligned}$$

From an economical perspective, the medium dimension is the main value-added model of distributors (e.g. radio or television), as they rely on creating and monetizing attention. That is, distributors provide information that attracts users in order to sell the gained attention to advertisers in order to monetize reproductions. In the context of electronic documents and the WWW, companies like Facebook that focus on the exploitation of human interaction rely on the medium dimension. (See Salaün and Collegium de Lyon 2011, Salaün 2012, passim)

In summary, the "vu, lu, su"-model describes any document as relying on the three dimensions of form (legibility—perception), content (understanding—assimilation), and medium (sociability—integration), that can be analyzed independently but also in context to each other. Table 2.1 on the next page gives an outline of different dimension-related research domains that examine electronic documents.

Approach	**Researchers and Scientists**	**Object** ***Results***	**Achievement** ***Interrogation***
form/sign (Vu)	Image Processors, Electronic Edition, W3C, Anthropology, History of Books	Carrier/Structure *Formats*	XML *Perception?*
text/content (Lu)	Linguistics, Information Sciences, Knowledge Engineering	Meaning/Meta *Processors*, *Ontologies*	Semantic Web *Knowledge?*
medium/relation (Su)	Telecommunication, Communication Studies, Management, Social Science	Collection/Hyper *Portals*, *Collaborations*, *Sites*	Library and Electronic Archives *Publication/* *Communication?*

Table 2.1.: *Domains covered by researches on the electronic document (cf. Pédauque 2007, Tab. 1, translated and reformatted by F. S.)*

2.3. Documentation, Document, Doceme

The documentation scientist Niels Windfeld Lund uses the notions of material, mental, and social phenomena as the three separate yet complementary aspects to describe a document. These aspects match with the previously described Pédauquian dimensions of form, content, and medium. But Lund adds two concepts to the notion of document that are again subject of analysis with regard to the three aspects: the documentation as the process of creating a document and the doceme as a distinct part of a document. These concepts are described in detail below. (See particularly Lund 2003a,b)

The *documentation* is described as the process that results in a coherent document. This process includes one or more human producers who use specific instruments (e.g. voice or fingers) in a specific way (so called modes) to create a document. Due to these parameters there can be different forms of documentation resulting in different documentation traditions that create specific kinds of documents—i.e., the different documentation traditions with their specific choices of physical and cognitive configurations and with their distinct social embedding create distinguishable types of documents that constitute distinguishable genres.

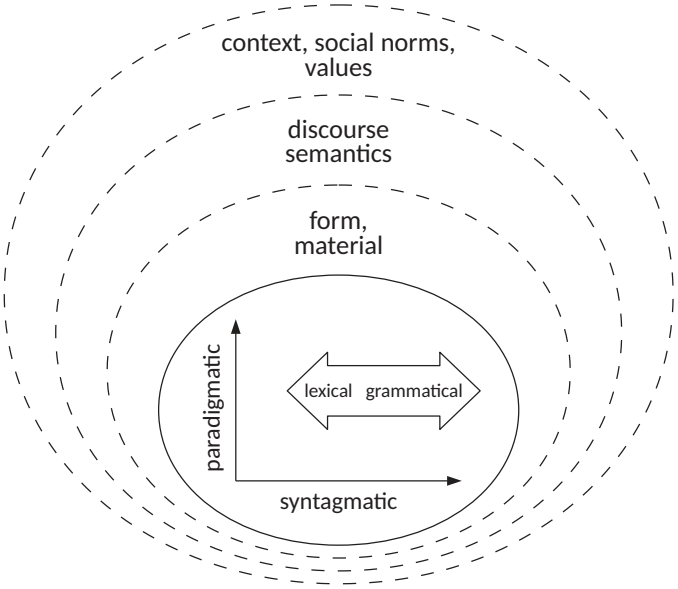


Figure 2.4.: *Semiotic mode model of a document and its stratal organization (cf. Bateman and Schmidt 2011, Fig. 3.3)*

The *doceme* is described as a part of a document that can be isolated as a partial result of documentation. This segmentation of the document can be used to analyze document parts on their own as well as in the context of other parts or the document as a total. Examples for docemes are pictures within articles or chapters within a book.

In summary: while the material, mental, and social phenomena differentiate the perspective of analyzing, the differentiation by documentation, document, and doceme states more precisely the analyzed item.

2.4. Layout, Rhetoric, Navigation

The linguist John Bateman describes a document as a multi-layered ("stratal") semiotic artifact to stress on the role of semiotic modes that are active in a document creation process (Bateman 2011). As sketched in Figure 2.4 above a semiotic mode is defined by a coding system that is composed of a non-material component and a material component that organizes specific information. This system creates a space of possible semiotic decisions over a paradigmatic and a syntagmatic axis to define the non-material component, where

the topology in this space is within a range from lexical to grammatical organization. Made decisions within this space result in structural configurations that leave as a consequence traces in the material form of information representation. In order to be interpretable the information described by means of a specific semiotic mode relies not only on the coding system, but has contextual dependencies as well. These dependencies often can be described through discourse semantics that offer rules of possible interpretations for a given information.

Bateman (2011) gives three examples of specific semiotic modes that can be active in visual page-based artifacts: text-flow, the mode active for written text; static image-flow, active for strip-like composed pictures or diagrams; page-flow, active for compositions in a two-dimensional space (e.g. composition of a page).

Based on this theoretical framework of semiotic modes, Bateman (2008) defines a document model consisting of five layers to distinguish between basic units, layout, rhetoric, navigation and genre. The higher layer's structures (layout, rhetoric, navigation) reference on basic physical units as their underlying elements—these *basic units* belong to the document's base layer. Different structures might need a different set and granularity of referable basic units that the base layer has to cope with. The totality of these units is what the physical document is composed of. The basic units are organized by means of the layout, rhetorical, and navigation layer as described below.

The *layout layer* describes the physical structure and realization of a document as it is sensorially perceived. For example, for a visual page-based document this layer would define segments that refer to basic units and that specify the visual characteristics (e.g. typography) and positioning within an area tree. This layer represents the material organization of a document.

The *rhetorical layer* describes the internal meaningful relationships and organization of a document by means of the Rhetorical Structure Theory (RST) (see Mann and Thompson 1988). A RST tree distinguishes core segments (nuclei) and side segments (satellites) that refer to basic units and that specifies the rhetorical relations between segments as e.g. evaluation (one segment is an evaluative comment on the other segment), cause (one segment describes the cause of the other segment) or restatement (one segment re-expresses the other segment) relations, etc.

The *navigation layer* refers to basic units that serve as navigational support to the reader of a document. The identified navigational elements are described as pointer sinks and sources that can be used from inside or outside of the document. For example, in visual page-based documents these navigational elements can be realized by means of page numbers or hyperlinks.

Finally, the *genre layer* as the highest layer of the model describes the configuration patterns within and between the underlying layers for a specific type of documents. This set of rules affects the document composition choices for content, structure, and context of use. For example, a given rule could determine that a specific rhetorical structure always comes in conjunction with a specific physical realization.

2.5. Conclusion

With the preceding sections three document models were introduced that agree with the assumption that a document is based on three distinctive yet complementary concepts. The first model (Pédauque 2006a, 2007) detaches the notion of document from its prominent ties to a physical object and introduces the three equal valued dimensions of form (legibility—perception), content (understanding—assimilation), and medium (sociability—integration) instead. These dimensions of a document can be analyzed independently but also in relation to each other. The second model (Lund 2003a,b) supports these three perspectives but its focus of analysis differentiates between the production process of a document (documentation), the document itself, and distinguishable, non-basic parts of a document (docemes). This differentiation describes the document production as being subject to distinct documentation traditions that make use of different instruments and different instrument modalities; in other words: a specific document is a member of a specific genre. The notion of genre leads to the third model (Bateman 2008, 2011) that establishes the genre as the highest layer of rules that influences the composition of a document according to its layout, rhetorical, and navigational structure. Again, each of these structures can be loosely assigned to one of the three Pédauquian dimensions. The layout structure describes the materiality, the form of a document; the rhetorical structure describes the internal meaning, the content of a document; the navigational structure describes the network of attention creation within a document. While the first and second correspondences match directly, the third needs further clarification on how the navigational structure matches the social aspect of a document. This

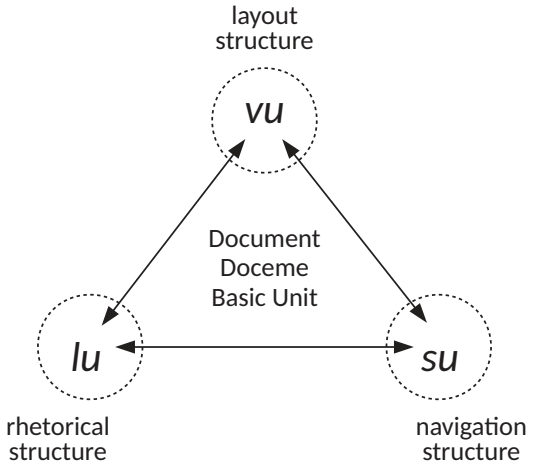


Figure 2.5.: *Genre-aware document model*

leads to the concept of Situation Theory (Barwise and Perry 1983, Devlin 1990) as applied to documents (Devlin and Rosenberg 1996, 2006, Lalmas 1998). An example is given by two tennis players that hit a ball towards each other. The interaction between both players only exists due to the ball—or more abstractly, a rally represents a flow of information that would not exist without the ball. Replacing the ball by a document leads to a similar situation, where the flow of information is induced by the document. A further example: one document refers to another by means of a quotation. Here, the trust in a primary resource is transferred to a secondary resource: the quoted part of the primary resource is typically identifiable in the document structure of the secondary resource and thus, a document and its navigational elements can be considered also as anchors of social interaction.

Following and combining the introduced document theories leads to the model sketched in Figure 2.5 above that distinguishes three segmentation types that are described from three exclusive and at the same time interacting perspectives. A document is segmented into basic units (u), docemes (d), and the complete document (D). The perspectives consist of the form (vu), content (lu), and medium (su) dimensions.

The *basic units* are the document elements that can be referenced, grouped, and organized by a layout tree, a RST tree, or a navigation path that implement the vu, lu, and su dimensions. Examples of dimension-related descriptions of a basic unit in text-centered documents could be a highlighted element (u_{vu}—e.g. bold typed), with a specific interpretation (u_{lu}—e.g. specific term), and a navigational impact (u_{su}—e.g. first appearance of indexed term).

The *doceme* is a recognizable substructure within a document. It is a subset of the layout tree, the RST tree, and the navigation path, respectively. Examples of docemes in a text-centered document could be a page-fragment that uses a distinctive output modality (d_{vu}—e.g. an image and its caption), that supports a main statement (d_{lu}—e.g. by means of an alternative restatement), and that constitutes a referenceable information portion (d_{su}—e.g. by means of a list of figures).

The *document* is the self-contained structure that consists of a layout tree, a RST tree, and a navigational structure. Each tree represents a specific perspective of the total document: the document's physical appearance (D_{vu}), the document's interpretable meaning (D_{lu}), and the document's anchors of attention (D_{su}). The different rules and restrictions in building the vu-, lu-, and su-structures for specific types of documents is what distinguishes different genres from each other.

The document dimensions provide specific perspectives that create different entries to different document description and processing domains (see Tab. 2.1, p. 12). The vu dimension represents the typical entry of document analysis (e.g. Tang et al. 1991), where a given layout (vu) is analyzed in order to recreate the logical content structure (lu) separately from navigational elements (su). The su dimension represents the typical entry of web structure mining (e.g. Liu 2007), where given networks of hyperlinks (su) are analyzed to extract associated contents (lu) and automatically generate new output documents (vu). The lu dimension represents the typical entry to document publishing, where an authored content (lu) is given a specific layout (vu) and made available (su). As this latter approach represents the perspective for the design of a multichannel publishing workflow, the lu dimension can be considered as the entry perspective of the thesis at hand. The main interest will be the creation of varying output documents that belong to the vu dimension.

The introduced informal model forms the basis of the further document description approach. Specifically, the following chapter adopts the idea of different document perspectives (form, content, medium) and segmentation (document, doceme, basic unit) that are restricted by a higher concept (genre) to formally define what we call a genre-aware document model.

Semanticists should be obstetricians, not coroners of programming languages.
— John C. Reynolds

3. An Implementable Formalism for Genres

With the introduction of a genre-aware document model (see Chap. 2) a document relies on different structures (form, content, medium) that organize common resource fragments (document, doceme, basic unit). Thus, a formal representation of documents by means of a single tree structure is not possible anymore (Plaice and Rowley 2003)—i.e., the common document processing tools that are based on tree processing cannot be used to describe and process genre-aware documents. In order to conceptualize an appropriate document description and processing, first, the informal model from Chapter 2 is translated into a formal model that in turn can be translated into a technical implementation in Chapter 4.

Section 3.1 introduces a formal definition of documents, docemes, and basic units that can be organized by any number of structures. This model constitutes what we call a multistructured document. Section 3.2 adds the concept of genre to a multistructured document so that the relations between the document's components can be described and restricted for a specific type of documents. Finally, Section 3.3 concludes the developed formal model of genre-aware documents.

3.1. Multistructured Documents

Electronic representations of documents as e.g. via XML usually rely on tree structures, i.e., Directed Acyclic Graphs (DAGs) with exactly one source and with totally ordered leafs. But the document model introduced in the previous Chapter 2 uses multiple perspectives and alternative content representations that exceed the characteristics of a tree. Formal approaches that cope with that problem define a document model by means of less restrictive graph structures (e.g. Bird and Liberman 2001, Marcoux et al. 2013). A particular formal model that considers a subsequent XML implementation as so-called multistructured documents

has been introduced by Abascal et al. (2004). This model defines the components of a multistructured document by means of a set of interlinked trees that describe the content or a document structure each.

In this Section 3.1 the informal model of documents with multiple perspectives as introduced in Chapter 2 is formally redefined step by step. This new model is based on the original model of multistructured documents (Abascal et al. 2004) and a revised formalization of content as a distinctive structure (Portier et al. 2012). The underlying models are summarized in Appendix A.2. The following Subsection 3.1.1 defines the content of the document by means of abstract basic units that are linked to alternative realizations each. Section 3.1.2 defines the structures of a document by means of ordered labeled trees. Section 3.1.3 defines how document structures and content bases are referred to each other by means of so-called correspondences. Section 3.1.4 describes how content bases, document structures, and correspondences jointly form multistructured documents and introduces the concept of docemes and catalogs as sub- and supersets of multistructured documents. Finally, Sections 3.1.5 and 3.1.6 conclude the new model with an instructive example and a discussion.

3.1.1. Content Bases

The content base of a document assembles all content fragments of a document. We use the following notations[1]:

- D denotes a document.
- $\mathrm{F}(D)$ denotes the necessarily non-empty set of all possible content fragments of a document D.
- $\Phi(D) = \{\varphi\colon\ (\mathrm{F}(D))^p \times \mathrm{F}(D) \mid p \geq 1\}$ is the set of all $(p+1)$-relations that relate a piece of content with other pieces.

By these means we can define a content base as follows:

Definition 3.1 (Content Base).[2] A *content base* is a set of abstract *basic units* that assemble

1. These notations originate from Portier et al. (2012), cf. App. A.2.2, p. 128.
2. The formal definition of content bases is derived from the original definition of basic structures from Portier et al. (2012), cf. Def. A.8, p. 129, where we replace the notion of composition nodes with basic units and add the concept of content modes.

alternative realizations (e.g. textual, graphical, or audiovisual) of a common *content node* and that are potentially shared by different document structures. The content base is formally a DAG with a single source that is described by

$$B = (V, E)\,.$$

$V = \{root\} \cup U \cup \mathrm{F}_{\mathrm{Base}}(D)$ is the set of vertices of B where

- *root* is the source node of B, i.e. each further node of the content base is part of a path with *root* as its starting node;
- U is the set of *basic units*, i.e. the content nodes that can assemble alternative realizations of a specific content such that each node $u \in U$ is the destination of exactly one edge with *root* as its starting node and labeled with a relation $\varphi \in \Phi(D)$ that prescribes which elements of $\mathrm{F}_{\mathrm{Base}}(D)$ are linked to u;
- $\mathrm{F}_{\mathrm{Base}}(D) \subset \mathrm{F}(D)$ is a set of all necessarily disjoint content fragments, i.e. all distinct *content realizations* of all specific content nodes, where each node $n \in \mathrm{F}_{\mathrm{Base}}(D)$ is the destination of one or more edges from basic units.

$E = \{(root, u) \mid u \in U\} \cup (\bigcup_{u \in U} \mathrm{real}(u))$ is the set of edges of B where

- $\mathrm{real}(u)$ with $u \in U$ is the set of all the edges from the basic unit u to one or more elements of $\mathrm{F}_{\mathrm{Base}}(D)$. Each of these edges is labeled with a string that associates each element of $\mathrm{F}_{\mathrm{Base}}(D)$ to a class. This classification may be used by a relation $\varphi \in \Phi(D)$ to choose from the elements of $\mathrm{F}_{\mathrm{Base}}(D)$.

To distinguish specific classes of content realizations called *content modes*, we introduce the following additional notations to denote a classification of realizations:

- $M = \{\text{text}, \text{image}, \text{audio}, \dots\}$ is the set of *content modes* that can be distinguished in a document D.
- $\mathrm{F}_m(D)$ with $m \in M$ is the subset of $\mathrm{F}_{\mathrm{Base}}(D)$ that belongs to the class of the specified content mode m.
- $\mathrm{real}_m(u)$ with $u \in U$ and $m \in M$ is the set of all m-labeled edges from the basic unit u to one or more elements of $\mathrm{F}_m(D)$.

In the context of this work, $\mathrm{F}_{\mathrm{Base}}(D)$ consists always of disjoint subsets of realizations of distinct modes. That is:

$$\mathrm{F}_{\mathrm{Base}}(D) = \bigcup_{m \in M}^{\cdot} \mathrm{F}_m(D).$$

For an example, see Section 3.1.5 “Content Base” on page 24.

3.1.2. Document Structures

A document structure describes a distinct view on a document’s content by means of a specific organization and is defined as follows:

Definition 3.2 (Document Structure).[3] A *document structure* S is a finite non-empty set of potentially labeled *structure nodes* N that form an ordered tree with potentially labeled *arcs* $A \subset N \times N$, such that

$$S = \langle N, A, L, l_N, l_A \rangle$$

where

- L is a finite set of labels.
- $l_N \colon N \times L$ is a relation that associates the nodes in N with labels in L.
- $l_A \colon A \times L$ is a relation that associates the arcs in A with labels in L.

For an example, see Section 3.1.5 “Document Structures” on page 26.

3.1.3. Correspondences

A correspondence describes a reference between elements of different document structures or between a document structure and the content base and is defined as follows:

3. The formal definition of document structures is a slightly adapted version of the original definition of documentary structures from Portier et al. (2012), cf. Def. A.9, p. 129.

Definition 3.3 (Correspondence).[4] A *correspondence* corr (G, G') between two graphs G and G' with vertices V and V' is a nonempty binary relation from the vertices of the first towards the vertices of the second graph, that is

$$\text{corr}\,(G, G') \subseteq \wp\,(V \times V') - \{\varnothing\}\,.$$

$\wp(V \times V')$ is the powerset of a set $V \times V'$. Also, V and V' are said to be in correspondence with each other.

In our model we use this notation to specify "horizontal" correspondences between nodes of different document structures and "vertical" correspondences between structure nodes and basic units. Therefore we have the two possibilities:

- $\text{corr}\left(S_i, S_j\right) = \wp\left(N_i \times N_j\right) - \{\varnothing\}$ is the set of possible correspondences between structure nodes of two document structures S_i and S_j with nodes N_i and N_j (see Def. 3.2). We call these correspondences *horizontal*.
- $\text{corr}\,(S, B) = \wp\,(N \times U) - \{\varnothing\}$ is the set of possible correspondences between structure nodes of a document structure S and basic units of a content base B (see Def. 3.1). We call these correspondences *vertical*.

For an example, see Section 3.1.5 "Correspondences" on page 28.

3.1.4. Multistructured Documents and Docemes

A multistructured document describes a content base with (potentially) alternative document structures that are referred to each other by means of correspondences. Thus, we can define a multistructured document as follows:

Definition 3.4 (Multistructured Document).[5] A *multistructured document* is a set of document structures that each rely on units of a common content base by means of correspondences. A multistructured document is formally defined by

4. The formal definition of correspondences is based on the original definition from Abascal et al. (2004), cf. Def. A.2, p. 126.
5. The formal definition of multistructured documents is based on the original definition from Portier et al. (2012), cf. Def. A.7, p. 128.

$$D = \langle B, \mathit{III}, C \rangle$$

where

- B is the *content base* of D (see Def. 3.1),
- $\mathit{III} = \{S_i \mid i = 1..n\}$ is the set of *document structures* of D (see Def. 3.2),
- $C = \left\{C_{ij} \in \text{corr}\left(G_i, G_j\right) \mid G_i \in \mathit{III}, G_j \in \mathit{III} \cup \{B\}\right\}$ is a set of relations between nodes of document structures III or the content base B called *correspondences* of D (see Def. 3.3).

As the correspondences in C link the document structures $S_i \in \mathit{III}$ with the content base B, the multistructured document D is composed of the vertices and edges from $S_i \in \mathit{III}$, B, and C and forms a connected, acyclic graph. This graph possesses as its only sources the document structures' roots and as its sink the content base.

For an example, see Section 3.1.5 "Multistructured Document" on page 28.

Furthermore, documents can be considered as non-monolithic artifacts that consist of smaller coherent fragments or that are used themselves as fragments by other documents. Therefore, we introduce the concept of docemes and catalogs as sub- and supersets of multistructured documents. In this way a doceme is a multistructured document that is used as a fragment by another multistructured document:

Definition 3.5 (Doceme). Let $D = \langle B, \mathit{III}, C \rangle$ be a multistructured document with its content base $B = (V, E)$, its set of document structures $\mathit{III} = \{S_i = \langle N_i, A_i, L_i, l_{N_i}, l_{A_i} \rangle \mid i = 1..n\}$, and its correspondences $C = \left\{C_j \mid j = 1..k\right\}$. Then

$$d = \langle B', \mathit{III}', C' \rangle$$

is a *doceme* that constitutes (partially) D with

- $B' = (V', E')$ where $V' \subseteq V, E' \subseteq E$ is a subgraph of B,
- $\mathit{III}' = \left\{S'_i = \left\langle N'_i, A'_i, L'_i, l'_{N_i}, l'_{A_i} \right\rangle \mid i = 1..n\right\}$ where $\mathit{III}' \subseteq \mathit{III}$ and each $N'_i \subseteq N_i, A'_i \subseteq A_i, L'_i \subseteq L_i, l'_{N_i} \subseteq l_{N_i}, l'_{A_i} \subseteq l_{A_i}$, i.e. each S'_i is a substructure of one S_i,
- $C' = \left\{C'_j \mid j = 1..k\right\}$ where $C' \subseteq C$ is a subset of C.

To provide a potentially recursive notation of docemes that allows to treat any document as a doceme of another document and conversely, an alternative notation that relies on indexes with a given index set I might be desirable, as in $D \supset \biguplus_{i \in I} D_i$. However, if docemes are considered terminal without any existing intermediate levels of document descriptions, we prefer the lowercase notation. Often a multistructured document D can be considered as a composition of a finite disjoint set of terminal docemes which means $D \supset \biguplus_{i \in I} d_i$.

Furthermore, we define a compilation of multistructured documents as follows:

Definition 3.6 (Catalog). In the special case where the multistructured document $D = \biguplus_{i \in \mathbb{N}^+} d_i$ is solely based on docemes (see Def. 3.5) without adding further content, structure, or correspondence, i.e. D is an aggregation of a finite disjoint set of other multistructured documents d_i, we call D a *catalog* (cf. Def. A.4–A.6 in App. A.2).

3.1.5. An Example

This section illustrates the use of the introduced model for multistructured documents by means of the English idiom "killing two birds with one stone". Its short form "two birds one stone" has its Japanese equivalent "一石二鳥" ("isseki ni chou") that is literally the same but uses a different ordering: "one stone two birds". So the example provides a common content ("KILLING 2 BIRDS WITH 1 STONE") with two alternative realization types (roman letters and Chinese, Japanese, and Korean (CJK) characters) that are organized alternatively depending on the output language (English or Japanese). In the following these characteristics are formally described by the introduced models of content base, document structure, and correspondence that compose a multistructured document.

Content Base

In this example we distinguish two basic units ("1 STONE", "2 BIRDS") from another that compose the document's content. Each basic unit has two alternative realizations that are distinguished by the used language (English and Japanese). Figure 3.1 on the next page shows the example document's content base that is formally defined as follows: the set of all disjoint content fragments in our example document D is

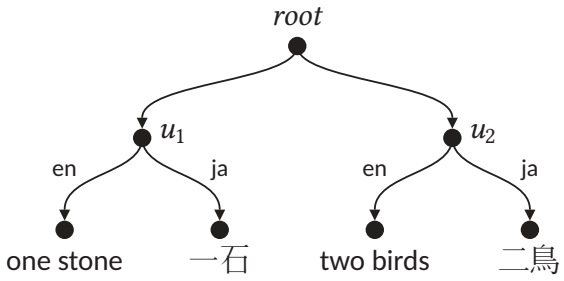


Figure 3.1.: *Graphical representation of the example content base*

$$\mathrm{F}_{\mathrm{Base}}(D) = \left\{\text{one stone, two birds, 一石, 二鳥}\right\}.$$

By means of these fragments we define the basic units u_1 with $\mathrm{real}(u_1) = \left\{(u_1, \text{one stone}), \left(u_1, 一石\right)\right\}$ and u_2 with $\mathrm{real}(u_2) = \left\{(u_2, \text{two birds}), \left(u_2, 二鳥\right)\right\}$, leading to the set of basic units $U = \{u_1, u_2\}$.

These constituent parts compose a graph such that

$$B = (V, E)$$

is the content base of document D where

- $V = \{root\} \cup \{u_1, u_2\} \cup \left\{\text{one stone, two birds, 一石, 二鳥}\right\}$ is the set of vertices of B,
- $E = \{(root, u_1), (root, u_2)\} \cup \left\{(u_1, \text{one stone}), \left(u_1, 一石\right), (u_2, \text{two birds}), \left(u_2, 二鳥\right)\right\}$ is the set of edges of B.

Additionally, we establish a classification of content realizations by using as content mode identifiers the set of distinguishable written languages denoted by their two-letter codes as standardized in ISO 639-1 (2002). Accordingly the English and Japanese languages are denoted by the strings “en” and “ja”. Note, that in this example we use exclusively CJK characters that represent pictograms or ideograms—i.e., the used CJK characters evolved from graphical representations of specific objects (e.g. stone or bird) and abstract ideas (e.g. one or two). Therefore, we could alternatively distinguish the content realizations e.g. by textual (roman letters) and graphical (CJK characters) mode instead of English and Japanese.

However, in our example we distinguish as follows:

- $M = \{\text{en}, \text{ja}\}$ is the set of written languages in document D.

Thus, each content fragment of the example is assigned to one (and only to one) specific language such that

$$\mathrm{F}_{\text{Base}}(D) = \mathrm{F}_{\text{en}}(D) \cup \mathrm{F}_{\text{ja}}(D)$$

is the set of all content fragments in document D distinguished by language where

- $\mathrm{F}_{\text{en}}(D) = \{\text{one stone}, \text{two birds}\}$ is the subset of content fragments in document D that are written in English,
- $\mathrm{F}_{\text{ja}}(D) = \{\text{一石}, \text{二鳥}\}$ is the subset of content fragments in document D that are written in Japanese.

Accordingly the edges from basic units to their specific realizations are distinguished by language identifiers. The example contains the following language-dependent sets of edges:

- $\text{real}_{\text{en}}(u_1) = \{(u_1, \text{one stone})\}$ is the edge from u_1 to its English realization "one stone" that is labeled as belonging to the content mode "en",
- $\text{real}_{\text{ja}}(u_1) = \{(u_1, \text{一石})\}$ is the edge from u_1 to its Japanese realization "一石" that is labeled with content mode "ja", respectively,
- $\text{real}_{\text{en}}(u_2) = \{(u_2, \text{two birds})\}$ is the edge from u_2 to its English realization "two birds" that is labeled with "en", respectively,
- $\text{real}_{\text{ja}}(u_2) = \{(u_2, \text{二鳥})\}$ is the edge from u_2 to its Japanese realization "二鳥" that is labeled with "ja", finally.

Document Structures

In this example we define two document structures to distinguish an English and a Japanese representation of our document. The two document structures describe two identical ordered trees with one root and two children each, but with different language labeling.

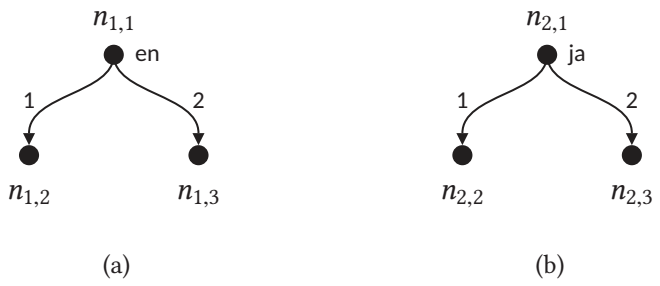


Figure 3.2.: *Graphical representation of the example document structures (a) S_1 and (b) S_2*

Figure 3.2 above shows the example document's structures that are formally defined such that

$$S_1 = \langle N_1, A_1, L_1, l_{N_1}, l_{A_1} \rangle \text{ and } S_2 = \langle N_2, A_2, L_2, l_{N_2}, l_{A_2} \rangle$$

are the two document structures of document D where

- $N_1 = \{n_{1,1}, n_{1,2}, n_{1,3}\}$ and $N_2 = \{n_{2,1}, n_{2,2}, n_{2,3}\}$ are the sets of structure nodes of S_1 and S_2,
- $A_1 = \{(n_{1,1}, n_{1,2}), (n_{1,1}, n_{1,3})\}$ and $A_2 = \{(n_{2,1}, n_{2,2}), (n_{2,1}, n_{2,3})\}$ are the sets of arcs of S_1 and S_2,
- $L = L_1 \cup L_2 = I \cup M = \{1, 2\} \cup \{\text{en}, \text{ja}\}$ is the set of labels of S_1 and S_2 containing natural numbers where I being a finite subset of $\mathbb{N}^+$ to give the structures an ordering and containing the documents' language codes to distinguish the document trees,
- l_{N_1}, l_{N_2} are the relations that associate to the roots of S_1 and S_2 a language code,
- l_{A_1}, l_{A_2} are the relations that associate to each edge a natural number and make S_1 and S_2 two ordered trees.

These constituent parts compose the document structures S_1 and S_2 with

- $S_1 = \langle \{n_{1,1}, n_{1,2}, n_{1,3}\}, \{(n_{1,1}, n_{1,2}), (n_{1,1}, n_{1,3})\}, \{1, 2, \text{en}\}, l_{N_1}, l_{A_1} \rangle$ where $l_{N_1} = \{(n_{1,1}, \text{en})\}$ and $l_{A_1} = \{((n_{1,1}, n_{1,2}), 1), ((n_{1,1}, n_{1,3}), 2)\}$,
- $S_2 = \langle \{n_{2,1}, n_{2,2}, n_{2,3}\}, \{(n_{2,1}, n_{2,2}), (n_{2,1}, n_{2,3})\}, \{1, 2, \text{ja}\}, l_{N_2}, l_{A_2} \rangle$ where $l_{N_2} = \{(n_{2,1}, \text{ja})\}$ and $l_{A_2} = \{((n_{2,1}, n_{2,2}), 1), ((n_{2,1}, n_{2,3}), 2)\}$.

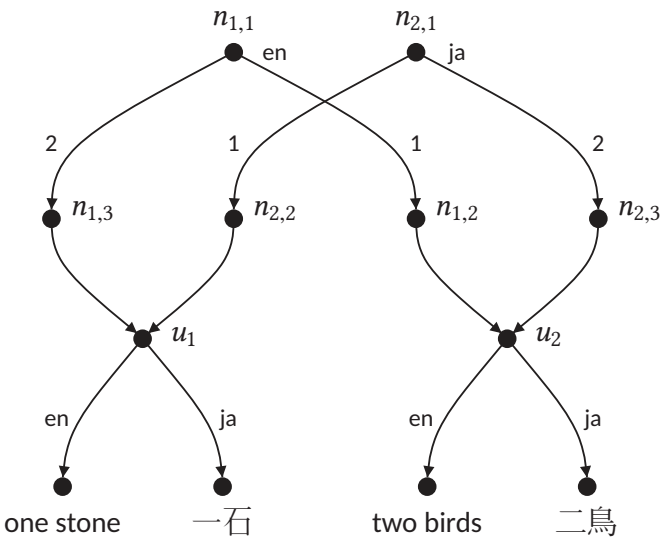


Figure 3.3.: *Graphical representation of the example document*

Correspondences

In this example we link the leafs of the English document structure S_1 such that its first leaf is linked to basic unit u_2 and its second leaf to basic unit u_1. The leafs of the Japanese document structure S_2 are linked to u_1 and u_2 inversely. These correspondences are formally defined as follows:

- $C = \{(n_{1,2}, u_2), (n_{1,3}, u_1)\} \cup \{(n_{2,2}, u_1), (n_{2,3}, u_2)\}$ is the set of correspondences between document structures S_1, S_2 and content base B.

Multistructured Document

In summary, the correspondences with its associated document structures and content base compose a multistructured document that describes the idiom "KILLING 2 BIRDS WITH 1 STONE". Figure 3.3 above shows the resulting graph that forms the multistructured document that is defined such that

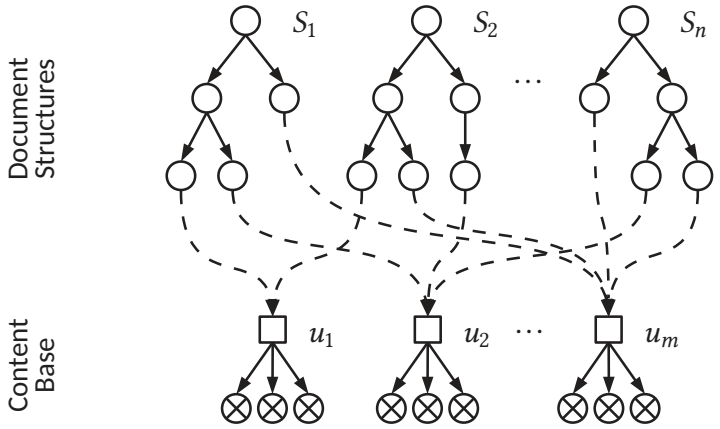


Figure 3.4.: *Graphical representation of a Multistructured Document with its Structure Nodes* (○)*, Basic Units* (□)*, Content Realizations* (⊗)*, Arcs* (→)*, and Correspondences* (⇢)

$$D = \langle B, \mathit{III}, C \rangle$$

where

- $B = \Big(\big\{ \mathit{root}, u_1, u_2, \text{one stone}, \text{two birds}, 一石, 二鳥 \big\},$
  $\big\{ (\mathit{root}, u_1), (\mathit{root}, u_2), (u_1, \text{one stone}), \big(u_1, 一石\big), (u_2, \text{two birds}), \big(u_2, 二鳥\big) \big\} \Big)$
  is the content base of D,
- $\mathit{III} = \{S_1, S_2\}$
  $= \Big\{ \Big\langle \{n_{1,1}, n_{1,2}, n_{1,3}\}, \{(n_{1,1}, n_{1,2}), (n_{1,1}, n_{1,3})\}, \{1, 2, \text{en}\}, l_{N_1}, l_{A_1} \Big\rangle,$
  $\Big\langle \{n_{2,1}, n_{2,2}, n_{2,3}\}, \{(n_{2,1}, n_{2,2}), (n_{2,1}, n_{2,3})\}, \{1, 2, \text{ja}\}, l_{N_2}, l_{A_2} \Big\rangle \Big\}$
  are the document structures of D,
- $C = \{(n_{1,2}, u_2), (n_{1,3}, u_1), (n_{2,2}, u_1), (n_{2,3}, u_2)\}$ are the correspondences of D.

3.1.6. Discussion

The previous sections have introduced a formal model that describes a multistructured document by means of a DAG. Figure 3.4 above summarizes the model with its document

structures and content base that are referred to each other by means of correspondences. Specifically, each document structure is composed of a labeled tree using nodes from the content base as its leaves. Therefore, distinctive substructures from the modeled graph are traversable via tree-based algorithms (as used in XML processing). Or more specific, distinct document structures can be processed via common XML tools and transformed to common XML representations again.

While the description of multistructured documents can strongly rely on preliminary work (especially Abascal et al. 2004, Portier et al. 2012), a formal distinction of types of documents by definition of distinct genres is still undone. Therefore, the following section adds to the formal model of multistructured documents a further formalism to distinguish a document by genre.

3.2. Genre-aware Documents

The previously introduced model of multistructured documents provides a formalism to define documents with multiple annotation structures for a content base as one self-contained graph. In this section a formalism to describe the membership of a document to a genre is added to the model of multistructured documents. As discussed in Chapter 2, a genre can be informally characterized by means of underlying construction rules that affect the document creation in terms of content (lu), form (vu), and context of use (su). These characteristics are now translated to formal patterns and restrictions for the content base, structures, and correspondence choices and compositions for a set of multistructured documents that comply with a shared genre. The formal definition of such genre-aware documents is developed in the following sections step by step.

Section 3.2.1 defines genre and subgenre to specify restrictions for the creation of a set of documents. Section 3.2.2 consolidates the models of genre and multistructured documents leading to a model of genre-aware documents. Finally, Sections 3.2.3 and 3.2.4 conclude the new model with an instructive example and a discussion.

3.2.1. Genres and Subgenres

A genre describes the common characteristics of a finite set of documents $\Delta = \{D_i \mid i = 1..n\}$ by means of their shared patterns and restrictions for content realizations, document structures, and correspondences. Before the concept of genre is formalized, we add to our notations for content fragments $\mathrm{F}(D)$ and content compositions $\Phi(D)$ (see Sec. 3.1.1) the following:[6]

- $\mathrm{F}_{\mathrm{Base}}(\Delta) = \bigcup_{D_i \in \Delta} \mathrm{F}_{\mathrm{Base}}(D_i)$ is the set of all content fragments of all documents $D_i \in \Delta$.
- $\Phi(\Delta) = \bigcup_{D_i \in \Delta} \Phi(D_i)$ is the set of all relations that relates a piece of content with other pieces for all documents $D_i \in \Delta$.

- $\mathrm{S}(D)$ denotes the set of all possible document structures of a document D.
- $\Sigma(D) = \{\sigma \colon (\mathrm{S}(D))^p \times \mathrm{S}(D) \mid p \geq 1\}$ is the set of all the $(p+1)$-relations that relate a structure with other structures of a document D.
- $\mathrm{S}(\Delta) = \bigcup_{D_i \in \Delta} \mathrm{S}(D_i)$ is the set of all document structures of all documents $D_i \in \Delta$.
- $\Sigma(\Delta) = \bigcup_{D_i \in \Delta} \Sigma(D_i)$ is the set of all relations that relate a structure with other structures for all documents $D_i \in \Delta$.

- $\mathrm{C}(D)$ denotes the set of all possible correspondences of a document D.
- $\mathrm{K}(D) = \{\kappa \colon (\mathrm{C}(D))^p \times \mathrm{C}(D) \mid p \geq 1\}$ is the set of all the $(p+1)$-relations that relate a correspondence with other correspondences of a document D.
- $\mathrm{C}(\Delta) = \bigcup_{D_i \in \Delta} \mathrm{C}(D_i)$ is the set of all correspondences of all documents $D_i \in \Delta$.
- $\mathrm{K}(\Delta) = \bigcup_{D_i \in \Delta} \mathrm{K}(D_i)$ is the set of all relations that compose a correspondence from other correspondences for all documents $D_i \in \Delta$.

For an example, see Sections 3.2.3 “Content Restrictions”, “Structure Restrictions”, and “Correspondence Restrictions” on pages 34 ff.

We can define a genre as follows:

6. The sets of the document’s main components (i.e. content fragments, document structures, correspondences) are denoted by a letter of the Roman alphabet (i.e. F, S, C) and its according sets of functions by the corresponding Greek letter (i.e. Φ, Σ, K).

Definition 3.7 (Genre). For a set of multistructured documents $\Delta = \{D_i \mid i = 1..n\}$ a genre G is a triple composed of content, structure, and correspondence restrictions as formally defined by

$$G = \left\langle \mathrm{F}_G(\Delta), \mathrm{S}_G(\Delta), \mathrm{C}_G(\Delta) \right\rangle$$

where

- $\mathrm{F}_G(\Delta)$ is the set of possible content realizations with its canonically associated set of realization choices $\Phi_G(\Delta)$ restricted by $\mathrm{F}_G(\Delta) \subseteq \mathrm{F}_{\mathrm{Base}}(\Delta)$ and $\Phi_G(\Delta) \subseteq \Phi(\Delta)$,
- $\mathrm{S}_G(\Delta)$ is the set of possible document structures with its canonically associated set of structure compositions $\Sigma_G(\Delta)$ restricted by $\mathrm{S}_G(\Delta) \subseteq \mathrm{S}(\Delta)$ and $\Sigma_G(\Delta) \subseteq \Sigma(\Delta)$,
- $\mathrm{C}_G(\Delta)$ is the set of possible correspondences with its canonically associated set of correspondence compositions $\mathrm{K}_G(\Delta)$ restricted by $\mathrm{C}_G(\Delta) \subseteq \mathrm{C}(\Delta)$ and $\mathrm{K}_G(\Delta) \subseteq \mathrm{K}(\Delta)$.

Note, if $\mathrm{F}_G(\Delta) \neq \mathrm{F}_{\mathrm{Base}}(\Delta)$ or $\mathrm{S}_G(\Delta) \neq \mathrm{S}(\Delta)$ or $\mathrm{C}_G(\Delta) \neq \mathrm{C}(\Delta)$, then the genre G is restrictive.

Furthermore, if Δ is clear from the context, we use the following abbreviations: F_G, S_G, and C_G describe $\mathrm{F}_G(\Delta)$, $\mathrm{S}_G(\Delta)$, and $\mathrm{C}_G(\Delta)$ as sets of possible content realizations, document structures, and correspondences with their canonically associated sets of realization choices and structure and correspondence compositions $\Phi_G(\Delta)$, $\Sigma_G(\Delta)$, and $\mathrm{K}_G(\Delta)$ as defined above.

Analogously to the relation between documents and docemes (see Def. 3.5), genres can be subdivided into derivative subgenres. In this way a subgenre is a genre that is derived from another, less restrictive genre:

Definition 3.8 (Subgenre). Let $G = \langle \mathrm{F}_G, \mathrm{S}_G, \mathrm{C}_G \rangle$ be a genre. Then

$$g = \left\langle \mathrm{F}_g, \mathrm{S}_g, \mathrm{C}_g \right\rangle$$

is a *subgenre* of G if

- $\mathrm{F}_g(\Delta) \subseteq \mathrm{F}_G(\Delta)$ is the derived set of possible content realizations and $\Phi_g(\Delta) \subseteq \Phi_G(\Delta)$ its canonically associated set of possible realization choices.
- $\mathrm{S}_g(\Delta) \subseteq \mathrm{S}_G(\Delta)$ is the derived set of possible document structures and $\Sigma_g(\Delta) \subseteq \Sigma_G(\Delta)$ its canonically associated set of possible structure compositions.
- $\mathrm{C}_g(\Delta) \subseteq \mathrm{C}_G(\Delta)$ is the derived set of possible correspondences and $\mathrm{K}_g(\Delta) \subseteq \mathrm{K}_G(\Delta)$ its canonically associated set of possible correspondence compositions.

To provide a potentially recursive notation of subgenres that allows to treat any genre as a subgenre of another genre and conversely, an alternative notation that relies on indexes with a given index set I might be desirable, as in $G \supset \bigcup_{i \in I} G_i$. However, if subgenres are considered terminal without any existing intermediate levels of genre descriptions, we prefer the lowercase notation. Often a genre G can be considered as a composition of a finite set of derivative terminal subgenres which means $G \supset \bigcup_{i \in I} g_i$.

3.2.2. Genre-aware Documents

A genre-aware document puts a multistructured document in context to other documents by means of a shared genre. More specifically, a genre-aware document is a multistructured document that complies with particular patterns and restrictions for its content, structure, and correspondence choices and compositions as specified by a specific genre. Thus, a genre-aware document is defined as follows:

Definition 3.9 (Genre-aware Document). A document $D = \langle B, \mathit{III}, C \rangle$ is compliant with a genre $G = \langle \mathrm{F}_G, \mathrm{S}_G, \mathrm{C}_G \rangle$ (see Def. 3.7), i.e. $D = D_G$ with

$$D_G = \langle B, \mathit{III}, C \rangle$$

if

- $B = (V, E)$ is the content base of D_G with its content realizations $\mathrm{F}_{\mathrm{Base}}(D_G) \subset V$ (see Def. 3.1) that are part of $\mathrm{F}_G(\Delta)$, i.e. $\mathrm{F}_{\mathrm{Base}}(D_G) \subseteq \mathrm{F}_G(\Delta)$,
- $\mathit{III} = \{S_i \mid i = 1..n\}$ are the document structures of D_G (see Def. 3.2) where $\mathit{III} \subseteq \mathrm{S}_G(\Delta)$,
- $C = \{C_i \mid i = 1..k\}$ are the correspondences of D_G (see Def. 3.3) where $C \subseteq \mathrm{C}_G(\Delta)$.

Furthermore, we informally call B, III, and C of a document $D_G = \langle B, \mathit{III}, C \rangle$ compliant with G, if $\mathrm{F}_{\mathrm{Base}}(D_G) \subseteq \mathrm{F}_G(\Delta)$, $\mathit{III} \subseteq \mathrm{S}_G(\Delta)$, or $C \subseteq \mathrm{C}_G(\Delta)$, respectively, and the context of these sets is clear. Analogously, a set Δ is compliant with a genre G, if all elements of Δ comply with G.

A document D is *genre-aware*, if it is compliant with at least one genre G, i.e. $D = D_G$ for at least one genre $G = \langle \mathrm{F}_G, \mathrm{S}_G, \mathrm{C}_G \rangle$.

3.2.3. An Example

This section describes an example genre using the previously introduced model of genre-aware documents. The example genre describes the Japanese four-character idioms called "Yojijukugo" (四字熟語) and the more specific subgenres of the idioms' English and Japanese representations. E.g. the idioms "KILLING 2 BIRDS WITH 1 STONE" from the preceding Section 3.1.5 and "MAKING 1 FORTUNE AT 1 STROKE" are realized by documents of the Yojijukugo genre. Correspondingly, the short forms "two birds one stone" and "a fortune one stroke" are members of the English Yojijukugo subgenre. Accordingly "一石二鳥" ("isseki ni chou"; literally "one stone two birds") and "一攫千金" ("ikkaku sen kin"; literally "one grasp thousand gold") are corresponding to realizations of the Japanese Yojijukugo subgenre.

So the example provides a common main genre G ("Yojijukugo") for a (starting) set of multistructured documents $\Delta = \{D_1, D_2\}$ ("KILLING 2 BIRDS WITH 1 STONE", "MAKING 1 FORTUNE AT 1 STROKE") that can be differentiated into further derivative subgenres g_{en} and g_{ja} (English or Japanese representation). In the following the genre and its subgenres are formally described by means of their introduced characteristics of patterns and restrictions for content configurations, document structures, and matching correspondences.

Content Restrictions

In this example we define the "Yojijukugo" genre such that it allows English and Japanese realizations in the content base:

- $\mathrm{F}_G(\Delta) = \left\{\text{one stone, two birds, 一石, 二鳥, one stroke, a fortune, 一攫, 千金, } \ldots\right\}$ is the set of all possible content realizations for a set of documents $D_i \in \Delta$ that are considered as compliant with our example genre G,
- $\varphi_G \colon U \times \mathrm{F}_G(\Delta) = \Big\{(u_1, \text{one stone}), \left(u_1, 一石\right), (u_2, \text{two birds}), \left(u_2, 二鳥\right),$
  $(u_3, \text{one stroke}), \left(u_3, 一攫\right), (u_4, \text{a fortune}), \left(u_4, 千金\right), \ldots\Big\}$
  is a relation $\varphi_G \in \Phi_G(\Delta)$ that associates all basic units $u_k \in U$ from $D_i \in \Delta$ with their possible content realizations within $\mathrm{F}_G(\Delta)$, where u_1, u_2 are assumed to be basic units of D_1 (see Section 3.1.5) and u_3, u_4 of D_2.

The preceding sets restrict the possible content realizations and realization choices within our example genre G that are further restricted in subgenres g_{en} and g_{ja} such that

$$\mathrm{F}_G \supseteq \mathrm{F}_{g_{en}} \cup \mathrm{F}_{g_{ja}}$$

are the language specific content restrictions of genre G where

- $\mathrm{F}_{g_{en}}(\Delta) = \{\text{one stone, two birds, one stroke, a fortune, } \ldots\}$ is the set of all possible content realizations within the English subgenre g_{en},
- $\mathrm{F}_{g_{ja}}(\Delta) = \left\{一石, 二鳥, 一攫, 千金, \ldots\right\}$ is the set of all possible content realizations within the Japanese subgenre g_{ja},

and

- $\varphi_{g_{en}} \colon U \times \mathrm{F}_{g_{en}}(\Delta) = \{(u_1, \text{one stone}), (u_2, \text{two birds}), (u_3, \text{one stroke}), (u_4, \text{a fortune}), \ldots\}$ is the relation that associates all basic units with their possible content realizations within the English subgenre g_{en},
- $\varphi_{g_{ja}} \colon U \times \mathrm{F}_{g_{ja}}(\Delta) = \left\{\left(u_1, 一石\right), \left(u_2, 二鳥\right), \left(u_3, 一攫\right), \left(u_4, 千金\right), \ldots\right\}$ is the relation that associates all basic units with their possible content realizations within the Japanese subgenre g_{ja}.

Structure Restrictions

In this example we define the "Yojijukugo" genre such that it allows to represent English, Japanese, and language independent document structures. Figure 3.5 on the next page shows the genre-compliant structures that are formally assembled as follows:

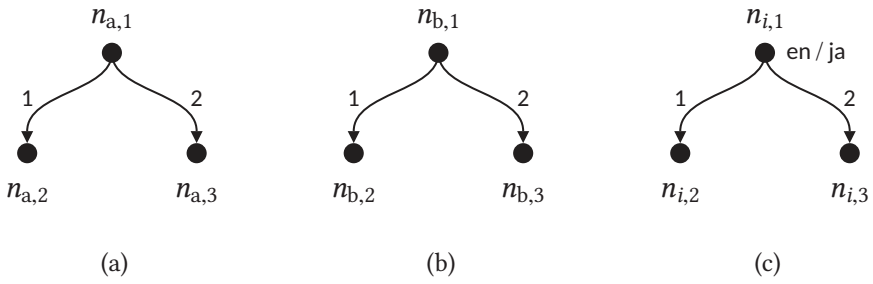


Figure 3.5.: *Graphical representation of the example document structures (a) S_a, (b) S_b, and (c) S_i*

- $\mathrm{S}_G(\Delta) = \{S_\mathrm{a}, S_1, S_2, S_\mathrm{b}, S_3, S_4, \dots\}$ is the set of all possible document structures for a set of documents $D_i \in \Delta$ that are members of our example genre G and where S_a, S_1, S_2 are structures of D_1 that describes the idiom "KILLING 2 BIRDS WITH 1 STONE" and S_b, S_3, S_4 of D_2 that describes "MAKING 1 FORTUNE AT 1 STROKE",
- $\sigma_G = \{(S_\mathrm{a}, S_1), (S_\mathrm{a}, S_2), (S_\mathrm{b}, S_3), (S_\mathrm{b}, S_4), \dots\}$ is a relation $\sigma_G \in \Sigma_G(\Delta)$ that has the intended interpretation to map from more general to more specific document structures for all $D_i \in \Delta$ within $\mathrm{S}_G(\Delta)$,

where

- $S_\mathrm{a} = \langle \{n_{\mathrm{a},1}, n_{\mathrm{a},2}, n_{\mathrm{a},3}\}, \{(n_{\mathrm{a},1}, n_{\mathrm{a},2}), (n_{\mathrm{a},1}, n_{\mathrm{a},3})\}, \{1, 2\}, \varnothing, l_{A_\mathrm{a}} \rangle$ with
  $l_{A_\mathrm{a}} = \{((n_{\mathrm{a},1}, n_{\mathrm{a},2}), 1), ((n_{\mathrm{a},1}, n_{\mathrm{a},3}), 2)\}$
  is a language independent document structure within $\mathrm{S}_G(\Delta)$ that belongs to D_1,
- $S_\mathrm{b} = \langle \{n_{\mathrm{b},1}, n_{\mathrm{b},2}, n_{\mathrm{b},3}\}, \{(n_{\mathrm{b},1}, n_{\mathrm{b},2}), (n_{\mathrm{b},1}, n_{\mathrm{b},3})\}, \{1, 2\}, \varnothing, l_{A_\mathrm{b}} \rangle$ with
  $l_{A_\mathrm{b}} = \{((n_{\mathrm{b},1}, n_{\mathrm{b},2}), 1), ((n_{\mathrm{b},1}, n_{\mathrm{b},3}), 2)\}$
  is a language independent document structure within $\mathrm{S}_G(\Delta)$ that belongs to D_2,
- $S_i = \langle \{n_{i,1}, n_{i,2}, n_{i,3}\}, \{(n_{i,1}, n_{i,2}), (n_{i,1}, n_{i,3})\}, L_i, l_{N_i}, l_{A_i} \rangle$ with
  $L_i = \{1, 2, \mathsf{en}\}$, $l_{N_i} = \{(n_{i,1}, \mathsf{en})\}$, $l_{A_i} = \{((n_{i,1}, n_{i,2}), 1), ((n_{i,1}, n_{i,3}), 2)\}$ for $i = 1, 3$,
  $L_i = \{1, 2, \mathsf{ja}\}$, $l_{N_i} = \{(n_{i,1}, \mathsf{ja})\}$, $l_{A_i} = \{((n_{i,1}, n_{i,2}), 1), ((n_{i,1}, n_{i,3}), 2)\}$ for $i = 2, 4$,
  are the language dependent document structures within $\mathrm{S}_G(\Delta)$ where S_1, S_3 describe English and S_2, S_4 Japanese document structures and where S_1, S_2 belong to D_1 and S_3, S_4 to D_2.

The preceding restrictions describe the possible document structures and structure compositions within our example genre G that are further restricted in subgenres g_{en} and g_{ja} such that

$$\mathrm{S}_G \supseteq \mathrm{S}_{g_{en}} \cup \mathrm{S}_{g_{ja}}$$

are the language specific structure restrictions of genre G where

- $\mathrm{S}_{g_{en}}(\Delta) = \{S_1, S_3, \dots\}$ is the set of all possible document structures within the English subgenre g_{en},
- $\mathrm{S}_{g_{ja}}(\Delta) = \{S_2, S_4, \dots\}$ is the set of all possible document structures within the Japanese subgenre g_{ja},

and

- $\sigma_{g_{en}} = \{(S_\mathrm{a}, S_1), (S_\mathrm{b}, S_3), \dots\}$ is the relation with the intended interpretation to map into all possible structures within the English subgenre g_{en} from language-independent structures,
- $\sigma_{g_{ja}} = \{(S_\mathrm{a}, S_2), (S_\mathrm{b}, S_4), \dots\}$ is the relation with the intended interpretation to map into all possible structures within the Japanese subgenre g_{ja} from language-independent structures.

Correspondence Restrictions

We define the "Yojijukugo" genre such that it allows the following correspondences between document structures and content base or other document structures:

- $\mathrm{C}_G(\Delta) = \{C_{\mathrm{a},0}, C_{\mathrm{b},0}, C_{1,0}, C_{2,0}, C_{3,0}, C_{4,0}, \dots\} \cup \{C_{\mathrm{a},\mathrm{b}}, C_{1,3}, C_{2,4}, \dots\}$ is the set of all possible vertical and horizontal correspondences of a set of documents Δ that are all members of our example genre G,
- $\kappa_G = \{(C_{\mathrm{a},0}, C_{1,0}), (C_{\mathrm{a},0}, C_{2,0}), (C_{\mathrm{b},0}, C_{3,0}), (C_{\mathrm{b},0}, C_{4,0}), \dots\} \cup \{(C_{\mathrm{a},\mathrm{b}}, C_{1,3}), (C_{\mathrm{a},\mathrm{b}}, C_{2,4}), \dots\}$ is a relation $\kappa_G \in \mathrm{K}_G(\Delta)$ that has the intended interpretation to map from more general to more specific correspondences for all $D_i \in \Delta$ within $\mathrm{C}_G(\Delta)$,

where

- $C_{\mathrm{a},0} = \{(n_{\mathrm{a},2}, u_1), (n_{\mathrm{a},3}, u_2)\}$, $C_{\mathrm{b},0} = \{(n_{\mathrm{b},2}, u_3), (n_{\mathrm{b},3}, u_4)\}$,
  $C_{1,0} = \{(n_{1,2}, u_2), (n_{1,3}, u_1)\}$, $C_{2,0} = \{(n_{2,2}, u_1), (n_{2,3}, u_2)\}$,
  $C_{3,0} = \{(n_{3,2}, u_4), (n_{3,3}, u_3)\}$, $C_{4,0} = \{(n_{4,2}, u_3), (n_{4,3}, u_4)\}$, …

with $C_{\mathrm{a},0} \in \mathrm{corr}\,(S_\mathrm{a}, B)$, $C_{\mathrm{b},0} \in \mathrm{corr}\,(S_\mathrm{b}, B)$, $C_{i,0} \in \mathrm{corr}\,(S_i, B)$ and $i \in \mathbb{N}^+$.
are assumed vertical correspondences within $\mathrm{C}_G(\Delta)$, where S_a, S_1, S_2 belong to D_1 and S_b, S_3, S_4 to D_2,

- $C_{\mathrm{a},\mathrm{b}} = \{(n_{\mathrm{a},1}, n_{\mathrm{b},1}), (n_{\mathrm{a},2}, n_{\mathrm{b},2}), (n_{\mathrm{a},3}, n_{\mathrm{b},3})\}$,
  $C_{1,3} = \{(n_{1,1}, n_{3,1}), (n_{1,2}, n_{3,2}), (n_{1,3}, n_{3,3})\}$,
  $C_{2,4} = \{(n_{2,1}, n_{4,1}), (n_{2,2}, n_{4,2}), (n_{2,3}, n_{4,3})\}, \ldots$
  with $C_{\mathrm{a},\mathrm{b}} \in \mathrm{corr}\,(S_\mathrm{a}, S_\mathrm{b})$, $C_{i,j} \in \mathrm{corr}\left(S_i, S_j\right)$ and $i, j \in \mathbb{N}^+$.
  are assumed horizontal correspondences within $\mathrm{C}_G(\Delta)$.

The preceding restrictions describe the possible vertical and horizontal correspondences and their compositions within our example genre G that are further restricted in subgenres g_{en} and g_{ja} such that

$$\mathrm{C}_G \supseteq \mathrm{C}_{g_{en}} \cup \mathrm{C}_{g_{ja}}$$

are the language specific correspondence restrictions of genre G where

- $\mathrm{C}_{g_{en}}(\Delta) = \{C_{1,0}, C_{3,0}, C_{1,3}, \ldots\}$ is the set of all possible correspondences within the English subgenre g_{en},
- $\mathrm{C}_{g_{ja}}(\Delta) = \{C_{2,0}, C_{4,0}, C_{2,4}, \ldots\}$ is the set of all possible correspondences within the Japanese subgenre g_{ja},

and

- $\kappa_{g_{en}} = \{(C_{\mathrm{a},0}, C_{1,0}), (C_{\mathrm{b},0}, C_{3,0}), (C_{\mathrm{a},\mathrm{b}}, C_{1,3}), \ldots\}$ is the relation with the intended interpretation to map into all possible correspondences within the English subgenre g_{en} from language-independent correspondences,
- $\kappa_{g_{ja}} = \{(C_{\mathrm{a},0}, C_{2,0}), (C_{\mathrm{b},0}, C_{4,0}), (C_{\mathrm{a},\mathrm{b}}, C_{2,4}), \ldots\}$ is the relation with the intended interpretation to map into all possible correspondences within the Japanese subgenre g_{ja} from language-independent correspondences.

Genre-aware Document

The "Yojijukugo" example document D_1 that describes the "KILLING 2 BIRDS WITH 1 STONE" idiom and that is compliant with $G = \langle \mathrm{F}_G, \mathrm{S}_G, \mathrm{C}_G \rangle$ constituting the "Yojijukugo" genre by means of content, structure, and correspondence restrictions as described above can be formally defined as follows:

$$D_{1,G} = \langle B, \mathit{III}, C \rangle$$

where

- $B = \Big(\big\{ root, u_1, u_2, \text{one stone}, \text{two birds}, 一石, 二鳥 \big\}, \big\{ (root, u_1), (root, u_2), (u_1, \text{one stone}), \big(u_1, 一石\big), (u_2, \text{two birds}), \big(u_2, 二鳥\big) \big\} \Big)$
  is the content base of D_1 that is compliant with G,
- $\mathit{III} = \Big\{ \big\langle \{n_{\mathrm{a},1}, n_{\mathrm{a},2}, n_{\mathrm{a},3}\}, \{(n_{\mathrm{a},1}, n_{\mathrm{a},2}), (n_{\mathrm{a},1}, n_{\mathrm{a},3})\}, \{1, 2\}, l_{N_\mathrm{a}}, l_{A_\mathrm{a}} \big\rangle, \big\langle \{n_{1,1}, n_{1,2}, n_{1,3}\}, \{(n_{1,1}, n_{1,2}), (n_{1,1}, n_{1,3})\}, \{1, 2, \mathsf{en}\}, l_{N_1}, l_{A_1} \big\rangle, \big\langle \{n_{2,1}, n_{2,2}, n_{2,3}\}, \{(n_{2,1}, n_{2,2}), (n_{2,1}, n_{2,3})\}, \{1, 2, \mathsf{ja}\}, l_{N_2}, l_{A_2} \big\rangle \Big\}$
  are the document structures of D_1 that are compliant with G,
- $C = \{(n_{\mathrm{a},2}, u_1), (n_{\mathrm{a},3}, u_2), (n_{1,3}, u_1), (n_{1,2}, u_2), (n_{2,2}, u_1), (n_{2,3}, u_2), (n_{\mathrm{a},1}, n_{\mathrm{b},1}), (n_{\mathrm{a},2}, n_{\mathrm{b},2}), (n_{\mathrm{a},3}, n_{\mathrm{b},3}), (n_{1,1}, n_{3,1}), (n_{1,2}, n_{3,2}), (n_{1,3}, n_{3,3}), (n_{2,1}, n_{4,1}), (n_{2,2}, n_{4,2}), (n_{2,3}, n_{4,3})\}$
  are the vertical and horizontal correspondences of D_1 that are compliant with G.

Correspondingly, the Japanese "Yojijukugo" subgenre compliant, more restrictive composition of document D_1 is formally described as follows:

$$D_{1,g_{ja}} = \langle B, \mathit{III}, C \rangle$$

where

- $g_{ja} = \big\langle \mathrm{F}_{g_{ja}}, \mathrm{S}_{g_{ja}}, \mathrm{C}_{g_{ja}} \big\rangle$ constitutes the Japanese subgenre of $G \supset g_{ja}$ as described above,
- $B = \Big(\big\{ root, u_1, u_2, 一石, 二鳥 \big\}, \big\{ (root, u_1), (root, u_2), \big(u_1, 一石\big), \big(u_2, 二鳥\big) \big\} \Big)$ is the content base of D_1 that is compliant with g_{ja},

- $III = \left\{\left\langle \{n_{2,1}, n_{2,2}, n_{2,3}\}, \{(n_{2,1}, n_{2,2}), (n_{2,1}, n_{2,3})\}, \{1, 2, \mathsf{ja}\}, l_{N_2}, l_{A_2}\right\rangle\right\}$ is the document structure of D_1 that is compliant with g_{ja},
- $C = \{(n_{2,2}, u_1), (n_{2,3}, u_2), (n_{2,1}, n_{4,1}), (n_{2,2}, n_{4,2}), (n_{2,3}, n_{4,3})\}$ are the vertical and horizontal correspondences of D_1 that are compliant with g_{ja}.

3.2.4. Discussion

The previous sections have introduced a formal definition of genres that extends the model of multistructured document by a genre-specific classification. This classification implies the restrictions for the composition of all genre compliant documents. Specifically, a genre-aware document is restricted in the content, structure, and correspondence choice and composition possibilities. As these restricted choices are formally described by sets of possible candidates and not by actual composition rules, the introduced genre definition has to be considered as a first informative approach, that provides orientation, not specification, for the subsequent implementation that is based on established validation and transformation techniques within the publishing industry (e.g. Schematron and XSLT, see Chap. 4).

3.3. Conclusion

With the preceding sections a formal document model was introduced that provides description options for alternative structures within a common document as well as description options for alternative realizations of a shared content fragment. Thus, these alternative document structures and realizations allow to describe the vu, lu, and su dimensions of a document (see Chap. 2) distinctively but within a common approach. With the formal definition of genre a concept has been introduced to characterize the relations between different structures and content realizations by means of composition restrictions. As an example, within a given genre a specific content structure usually comes in conjunction with a specific layout structure and a specific layout structure uses a specific content realization. In other words, specific composition restrictions are considered as typical for a specific type of document. As exemplified in Section 3.2.3, these restrictive relations can be described by means of functions that e.g. map a structure to a different structure. More specific, these functions describe how the vu-, lu-, and su-specific structures are transformed into one another within a given genre.

Thus, with the additional concept of subgenres, the model provides a concept to describe alternative transformations from a source structure to different output structures that describe different realizations of a single document. This concept is used in the following chapter to conceptualize and implement a multichannel publishing framework.

If I invent another programming language, its name will contain the letter X.

— Niklaus E. Wirth

4. A Technical Framework and its Implementation

With the formal definition of a genre-aware document model (see Chap. 3) a document relies on interlinked tree structures that describe a content base and its corresponding alternative organizations. These structures can comply with genre-specific restrictions that determine which structure compositions are allowed and how a given structure is transformed into a complementary structure within a document description. This approach is used in the following to build a multichannel publishing workflow by translating the formal definitions of the model into a technical implementation that relies on established standards of the publishing industry—i.e., XML-based formats and technologies as proposed by the W3C.

Specifically, Section 4.1 introduces a description framework for multistructured documents that is based on these established standards. Section 4.2 develops a processing framework for multistructured document descriptions that is based on established processing tools. Section 4.3 explains how to use the implemented framework to realize a multichannel workflow based on the concepts of genre and subgenre. Finally, Section 4.4 concludes the new framework.

4.1. Multistructured Document Description

The description of multistructured documents exceeds the markup possibilities for common XML documents that can only describe tree structures with one root. Furthermore, the combination of alternative document structures for a common content potentially leads to markup that cannot be described by a properly nested tree structure (Durusau and O'Donnell (2002), see Table 4.1 on the next page). The problem of applying overlapping and concurring markup to annotate XML content has been broadly discussed (e.g. Sperberg-McQueen and Huitfeldt 2004, DeRose 2004). Solutions range from using XML extensions

Relation	Symbol	Markup Example
A before *B*	$A \prec B$	<a>..... .....
A meets *B*	A m B	<a>..... .....
A overlaps *B*	A o B	<a>..... .....
A starts *B*	A s B	<a>..... .....
A during *B* [1]	A d B	<a>..... .....
A finishes *B*	A f B	<a>..... .....
A after *B*	$A \succ B$	<a>..... .....
A met-by *B*	A mi B	<a>..... .....
A overlapped-by *B*	A oi B	<a>..... .....
A started-by *B*	A si B	<a>..... .....
A contains *B*	A di B	<a>..... .....
A finished-by *B*	A fi B	<a>..... .....
A equal *B*	$A \equiv B$	<a>..... .....

Table 4.1.: *Possible structural relationships between concurring markup (cf. Allen 1983, Durusau and O'Donnell 2002)*

that allow non-wellformed element nesting or that extend the markup capabilities to mix concurring document types' markup within one document instance (see e.g. the "concur" feature introduced by ISO 8879 1986), to replacing common XML element annotations altogether with pairs of empty elements that mark the elements' boundaries (see e.g. the "milestone" method introduced by Burnard and Bauman 2015, Sec. 20.2).

To describe multistructured documents containing multiple media types like images, audio, or video, the separation of the markup from its content by using "standoff markup" (Thompson and McKelvie 1997) is a recognized solution. In contrast to the common inline XML markup where the document structure and its content nodes are conjointly represented within the markup, the standoff approach uses references to associate content nodes with their

1. The name of the "during" relation seems misplaced due to its time-related nature. While e.g. "within" might be a more suitable name in the context of overlapping markup, we keep the original notion to remain true to its origins (see Allen 1983).

respective annotations. Sophisticated standoff markup applications usually provide an underlying document model that describes graphs of nodes, edges, and respective labels that distinguish the alternative document structures, their structure annotations, and a content base (cf. Chapter 3). These formal models usually define their own proprietary markup representations that typically need their own processing tools (e.g. ISO 24612 2012, Chatti et al. 2007, Stührenberg and Jettka 2009). But, there are also examples of document description languages that make use of established XML standards to embed standoff markup mechanisms in their host languages: E.g. the TEI uses the XML Inclusions (XInclude) and XML Pointer Language (XPointer) standards to reference from defined document structures to according content sources (Burnard and Bauman 2015, Sec. 16.9, Marsh et al. 2006, 2003). Likewise the Corpus Encoding Standard for XML (XCES) uses the XML Linking Language (XLink) and XPointer standards (Ide et al. 2000, Maler et al. 2010).

An approach that defines a generic description language for multistructured documents by using established XML standards (XLink and XPointer) is provided by the Potsdamer Austauschformat Linguistischer Annotationen (PAULA)[2] XML format (Dipper and Götze 2005). PAULA does not only separate the document structures from the content, but deconstructs the document structures themselves by separating the annotation structures and their annotation "features". Thus, a specific structure not only refers to according content nodes but to its according labels as well. This allows to conceptualize a document processing framework that is based on the generic description of documents in general rather than on the specific participating document types. While this approach could be completely implemented by means of common XML standards, PAULA's designers have chosen to rely on their own generic host language. To overcome this last limitation, we introduce in the following a XLink-based framework to describe multistructured documents in a generic way that uses exclusively standards of the XML processing ecosystem.

The following Section 4.1.1 gives a general introduction to the XLink standard and its incorporation of the Resource Description Framework (RDF). Sections 4.1.2–4.1.5 introduce XLink expressions to distinguish document structuring and labeling—specified by means of the XML Schema model—from alternative content representations within a document graph. Finally, Sections 4.1.6 and 4.1.7 conclude this new approach with an instructive example and its discussion.

2. "Potsdam Exchange Format for Linguistic Annotations"

```
<link xlink:type="simple" xlink:href="scheme://path/file.txt">Link</link>
```

Listing 4.1.: *Simple XLink link example*

```
<link xlink:type="extended">
  <arc xlink:type="arc" xlink:from="source" xlink:to="sink"/>
  <res xlink:type="resource" xlink:label="source">Link</res>
  <res xlink:type="locator"  xlink:label="sink" xlink:href="scheme://path/file.txt"/>
</link>
```

Listing 4.2.: *Extended XLink link example*

4.1.1. XLink-based Document Description

The XML Linking Language (XLink) as described by Maler et al. (2010) defines a set of XML attributes to describe links between information nodes—so called "resources". These resources can be provided within the link description itself (i.e. local) or by a reference (i.e. remote). Unlike typical hyperlinks in the context of the WWW that point from one local resource to one remote resource, XLink can describe multidirectional links between multiple local and remote resources. Information about direction, behavior, and role of link traversal is attached to "arcs". Especially, XLink allows to describe outgoing and incoming edge lists—i.e. arcs that have local starting and remote ending resources (so-called "outbound arcs") and vice versa (so-called "inbound arcs"). The special case of a collection of links that contain only remote starting resources is called a "linkbase". A distinctive feature of XLink is the integratability of these link descriptions with other XML-based host languages. This provides a universal linking mechanism to XML in general.

Besides the full support of this so called "extended" linking mechanisms, XLink provides a simplified markup that is a redefinition of common WWW hyperlinks, also. Listing 4.1 above shows an example definition of a simple-type link, that connects the local resource "`Link`" with the remote resource referenced by the `xlink:href` attribute. Listing 4.2 above shows the same link described as an extended-type link. A more complex link can be defined by adding further arcs and resources to the XLink definition.

Furthermore, XLink incorporates the possibility of semantic markup to describe the meaning of resources within a link and their relations to each other. Derived from the Resource

	notion	RDF Schema 1.1[3]	meaning
structuring	Bag	`#Bag`	The starting-resource contains the (unordered) set of specified ending-resources
	Sequence	`#Seq`	The starting-resource contains the (ordered) sequence of specified ending-resources
	Alternative	`#Alt`	The starting-resource contains as alternatives the specified ending-resources
labeling	Type	`#type`	The starting-resource is an instance of a class specified by the ending-resource
	Value	`#value`	The starting-resource has the idiomatic property specified by the ending-resource

Table 4.2.: *RDF Schema data-modeling vocabulary*

Description Framework (RDF) as descibed by Wood et al. (2014), each resource and arc can be annotated with an Internationalized Resource Identifier (IRI) that denotes a semantic concept forming together sets of statements, which informally can be represented by the following triple[4]:

⟨*starting-resource*⟩ ⟨has *arc-role*⟩ ⟨*ending-resource*⟩ .

RDF itself provides a collection of IRIs specifying a semantic core vocabulary (Guha and Brickley 2014). Table 4.2 above shows the subset of the RDF core vocabulary that is used in the following to annotate XLink arcs to relate resources semantically to each other. Additionally, XML Schema (Maloney et al. 2004) provides simple and complex data type definitions within its own schema representation. These types are referenceable by unique identifiers and partially promoted as built-in data types for XML in general (see Biron and Malhotra 2004). Table 4.3 on the next page shows the types that are used in the following to define XML structures via XLink resource annotations. A full, uncommented list of the fragment identifiers in the XML Schema implementation of XML Schema itself is given in Appendix A.3.

3. The RDF fragment identifiers refer to `http://www.w3.org/1999/02/22-rdf-syntax-ns` .
4. Note that in this chapter the representations of a RDF tuple are loosely based on but not strictly conforming to the Terse RDF Triple Language (Turtle) syntax (Carothers and Prud'hommeaux 2014).

	XML 1.0	XML Schema 1.0[5]	meaning
node type	Element	`#element`	XML element
	Attribute	`#attribute`	XML attribute
naming pattern	`Name`	`#Name`	XML name
	`NCName`	`#NCName`	unqualified (non-colonized) XML name; e.g. `div`
	`QName`	`#QName`	qualified (prefixed) XML name; e.g. `html:div`
attribute value type	`CDATA`	`#string`	character string
	`ID`	`#ID`	unique identifier
	`IDREF`	`#IDREF`	reference to a unique identifier
	`IDREFS`	`#IDREFS`	list of references to unique identifiers
	`ENTITY`	`#ENTITY`	external entity
	`ENTITIES`	`#ENTITIES`	list of external entities
	`NMTOKEN`	`#NMTOKEN`	name token
	`NMTOKENS`	`#NMTOKENS`	list of name tokens
	`NOTATION`	`#NOTATION`	notation identifier

Table 4.3.: *DTD and XML Schema types (cf. Marchal 2002, Appx. B)*

Thus, XLink allows markup to describe graphs consisting of nodes and edges that can be annotated with appropriate labels. For graph descriptions as outgoing or incoming edge lists or as a mixture of both, XLink can realize each of these ways of description. In the following sections outgoing edge lists will be used for document structuring and incoming edge lists for document labeling. The special case of an edge list where initial and terminal nodes are both described as remote resources will be used for basic units.

4.1.2. Content Base

In the following, a content base is described by an edge list, where each list item describes an abstract basic unit that refers to alternative specific content nodes. These content

5. The XML Schema fragment identifiers refer to `http://www.w3.org/2001/XMLSchema` .

content mode	architectures	DTD's system identifier / namespace IRI [6]
Text	*diverse*	`http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod`
List	`<ul>,<ol>,<dl>`	`http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod`
Table	`<table>`	`http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod`
Image	`<img>`	`http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod`
Audio	`<audio>`	`http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod`
Video	`<video>`	`http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod`
...	*further*	...

Table 4.4.: *Content modes and their XHTML implementation*

nodes describe the same content portions in different modalities. Examples for different content modes are full text, table, image, audio, video, or other architectures e.g. for three-dimensional realizations. The content base can be described via XLink as a linkbase that is associated to the documents that contain the specific contents.

Each list item in this linkbase is described as a link that specifies the alternative content realizations of one basic unit as remote resources that refer to the specific realizations. The corresponding modality of each realization is described as semantic annotation by an IRI. Some modalities can be distinguished e.g. by Extensible HTML (XHTML) modules (Austin et al. 2010) as shown in the upper part of Table 4.4 above. Each content base's link thus intentionally represents the following RDF statement:

$$\langle node_i \rangle \quad \langle \texttt{has alternative} \rangle \quad (\langle node_1 \rangle \texttt{ , ... , } \langle node_n \rangle) \texttt{ .}$$

An example that shows a XLink definition of a basic unit with two alternative content realizations is given in Listing 4.3 on the next page. The example embeds the link in a `li` element and its specific definitions in subordinated `span` elements. The arc defined in line 2 specifies the ending resources as alternative (`arcrole="#Alt"`, cf. Tab. 4.2, p. 46) replacements (utilizing `show="replace"`) to the starting resource. As each resource is associated to all resources within the link, the `from` and `to` attributes are omitted in the arc specification, as

6. Note that the IRIs for text, list, table, and image modes refer to actual DTD descriptions of XHTML modules (Austin et al. 2010), those for audio and video do not, but stand for distinct XHTML 5 elements (Berjon et al. 2014).

```
1 <li id="unit_1" xlink:type="extended">
2   <span xlink:type="arc" xlink:show="replace"
        xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
3   <span xlink:type="locator" xlink:label="_" xlink:href="#text_1"
        xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
4   <span xlink:type="locator" xlink:label="_" xlink:href="#image_1"
        xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod"></span>
5 </li>
```

Listing 4.3.: *XLink definition of an example basic unit and its alternative representations*

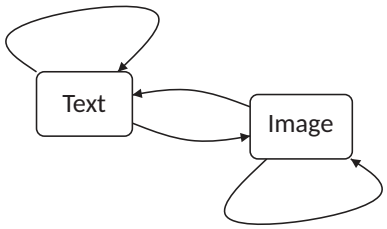


Figure 4.1.: *Graphical representation of the example basic unit with its content realizations* (▭) *and connecting arcs* (→) *as defined in Listing 4.3 above*

this is interpreted as standing for all supplied XLink labels within the link (here the "_" label). The textual realization of basic unit $\texttt{unit}_1$ is defined in line 3 and its graphical realization in line 4. Thus, the example link canonically represents the following RDF statements: $\langle\texttt{text}_1\rangle$ ⟨has alternative⟩ ($\langle\texttt{text}_1\rangle$,$\langle\texttt{image}_1\rangle$) and $\langle\texttt{image}_1\rangle$ ⟨has alternative⟩ ($\langle\texttt{text}_1\rangle$, $\langle\texttt{image}_1\rangle$). The described graph is shown in Figure 4.1 above.

4.1.3. Document Structuring

The specific document structure is described by an outgoing edge list, where each list item describes a structure node and the edges to its sequentially ordered child nodes.

A list item can be described via XLink as a link that defines the node as a local resource and the edges to its child nodes as remote resources. These can refer to other list items of the document structure or to basic units, i.e. content nodes. A document structuring link thus intentionally represents the following RDF statement:

```
1 <li xlink:type="extended">
2   <span xlink:type="arc" xlink:from="init" xlink:to="ter" xlink:show="embed"
        xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
3   <span id="node_1"  xlink:type="resource" xlink:label="init"></span>
4   <span id="edge_1.1" xlink:type="locator" xlink:label="ter" xlink:href="#node_1.1"></span>
5   <span id="edge_1.2" xlink:type="locator" xlink:label="ter" xlink:href="#node_1.2"></span>
6 </li>
```

Listing 4.4.: *XLink definition of an example structure node and its outgoing edges*

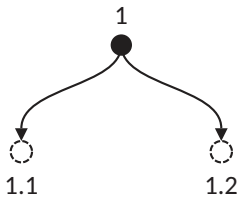


Figure 4.2.: *Graphical representation of the example structure node with its local* (●) *and remote* (◌) *resources and its connecting arcs* (→) *as defined in Listing 4.4 above*

$$\langle node_x \rangle \quad \langle \texttt{has sequence} \rangle \quad (\langle node_{x.1} \rangle \ , \ \langle node_{x.2} \rangle \ , \ \ldots \ ) \ .$$

An example that shows a XLink definition of a structure node with two children is given in Listing 4.4 above. The example embeds the link in a `li` element and its specific definitions in subordinated `span` elements, where the structure node is defined in line 3 and its two outgoing edges in line 4 and 5. The connecting arc defined in line 2 adds the ending resources as sequentially ordered (`arcrole="#Seq"`, cf. Tab. 4.2, p. 46) and subordinated children (utilizing `show="embed"`) to the starting resource. Thus, the example link canonically represents the following RDF statement: $\langle \texttt{node}_1 \rangle \ \langle \texttt{has sequence} \rangle \ (\langle \texttt{node}_{1.1} \rangle, \langle \texttt{node}_{1.2} \rangle)$. The described graph is shown in Figure 4.2 above.

4.1.4. Document Labeling

The document labeling is described by an incoming edge list, where each list item describes a label node and the edges that point from structure nodes to it.

```
1 <li xlink:type="extended">
2   <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
        xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
3   <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1"
        xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
4   <span xlink:type="locator" xlink:label="elem" xlink:href="#node_2"
        xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
5   <span xlink:type="resource" xlink:label="type"
        xlink:role="http://www.w3.org/2001/XMLSchema#QName"><namespace-name:local-part></span>
6 </li>
```

Listing 4.5.: *XLink definition of an example element label and its incoming edges*

```
1 <xsl:element name="{$element-type}">
2   <xsl:apply-templates mode="process-attribute-labels"/>
3   <xsl:apply-templates mode="process-content"/>
4 </xsl:element>
```

Listing 4.6.: *XSLT processing of an element label*

Such a list can be realized via XLink as a linkbase that is related to its associated document structure. Each list item in the linkbase is described by a link defining label nodes as local resources and the edges to corresponding structure nodes as remote resources. As XML defines different node types (e.g. elements, comments, attributes), the specific labeling type is given as a semantic annotation to the remote starting resource. XML provides different naming patterns to specify node labels (in particular unqualified and qualified names) and different value types for attribute values (e.g. ‘character string’ or ‘name token’) that are given as semantic annotations to the local ending resources. Table 4.3 on page 47 shows XML Schema IRIs that can be used as semantic annotations accordingly. In the following we will only refer to element and attribute nodes.

Element Labels

XML documents are primarily structured by element nodes of distinct element types. These element types can be distinguished by a naming pattern that consists of a namespace and a local name, forming together a so called “qualified name” (Thompson et al. 2009). This

naming pattern is used to specify element labels via XLink links that intentionally represent the following RDF statement:

⟨element::node⟩ ⟨has type⟩ ⟨QName::*elem-type*⟩ .

An example that shows a XLink definition of an element label for two associated structure nodes is given in Listing 4.5 on the preceding page. The example embeds the link in a `li` element and its specific definitions in subordinated `span` elements. The arc defined in line 2 adds the ending resource as instantiation type (`arcrole="#type"`, cf. Tab. 4.2, p. 46) to the starting resources. Two incoming edges are defined by the starting resources in line 3 and 4. They prepare the referenced resources $\texttt{node}_1$ and $\texttt{node}_2$ for being labeled as an element (`role="#element"`). The specific element type is described in the ending resource by means of a qualified naming pattern (`role="#QName"`) in line 5. Thus, the example link canonically represents the following RDF statements: ⟨element::$\texttt{node}_1$⟩ ⟨has type⟩ ⟨QName::*namespace-name:local-part*⟩ , ⟨element::$\texttt{node}_2$⟩ ⟨has type⟩ ⟨QName::*namespace-name:local-part*⟩. Listing 4.6 on the preceding page shows the intended XML element creation by means of a conceptually compatible XSL Transformation (XSLT) definition.

Attribute Labels

Each element node can be further labeled by attribute annotations. Unlike element labels, attribute labels are type-value pairs that can be specified via XLink links that intentionally represent the following RDF statement:

⟨attribute::node⟩
⟨has type⟩ ⟨QName::*attr-type*⟩ ;
⟨has value⟩ ⟨string::*attr-value*⟩ .

An example that shows a XLink definition of an attribute label for one corresponding structure node is given in Listing 4.7 on the next page. The example embeds the link in a `li` element and its specific definitions in subordinated `span` elements. The arcs defined in lines 2 and 3 add the ending resources as instantiation type and value (`arcrole="#type"`, `arcrole="#value"`, cf. Tab. 4.2, p. 46) to the starting resource. The incoming edge defined by the starting resource in line 4 prepares the referenced resource $\texttt{node}_1$ for being labeled as an attribute (`role="#attribute"`). The specific attribute type and value are defined in

```
1 <li xlink:type="extended">
2   <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
        xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
3   <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
        xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
4   <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1"
        xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
5   <span xlink:type="resource" xlink:label="type"
        xlink:role="http://www.w3.org/2001/XMLSchema#QName">namespace-name:local-part</span>
6   <span xlink:type="resource" xlink:label="value"
        xlink:role="http://www.w3.org/2001/XMLSchema#string">value</span>
7 </li>
```

Listing 4.7.: *XLink definition of an example attribute label and its incoming edge*

```
1 <xsl:attribute name="{$attribute-type}" select="$attribute-value"/>
```

Listing 4.8.: *XSLT processing of an attribute label*

the ending resources by means of a qualified naming pattern (`role="#QName"`) and a string (`role="#string"`) in lines 5 and 6. Thus, the given link canonically represents the following RDF statements: ⟨`attribute::node`$_1$⟩ ⟨`has type`⟩ ⟨`QName::`*`namespace-name:local-part`*⟩ , ⟨`attribute::node`$_1$⟩ ⟨`has value`⟩ ⟨`string::`*`value`*⟩. Listing 4.8 above shows the intended XML attribute creation by means of a conceptually compatible XSLT definition.

4.1.5. Multistructured Documents

All components of a specific multistructured document (see Sec. 4.1.2–4.1.4) are put in relation to each other in a single XLink link. Here each document structure is associated with a linkbase that contains the corresponding label nodes (cf. preceding Sec. 4.1.4). The global labeling type (i.e. XML document type) of each document structure is specified by a namespace IRI as semantic annotation to the linkbase reference. Optionally a further arc can associate the specific content documents with the basic units' linkbase. As the basic units are referenced from within the structures this information is not intended for processing, but provided for the sake of clarity.

```
1  <div xlink:type="extended">
2    <!-- Structure 1 -->
3    <span xlink:type="arc" xlink:from="s1" xlink:to="l1" xlink:show="embed"
           xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"></span>
4    <span xlink:type="locator" xlink:label="s1" xlink:href="S1.struct.xml#node_1"></span>
5    <span xlink:type="locator" xlink:label="l1" xlink:href="S1.labels.xml"
           xlink:role="http://www.w3.org/1999/xhtml"></span>
6    <!-- Structure 2 -->
7    <span xlink:type="arc" xlink:from="s2" xlink:to="l2" xlink:show="embed"
           xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"></span>
8    <span xlink:type="locator" xlink:label="s2" xlink:href="S2.struct.xml#node_1"></span>
9    <span xlink:type="locator" xlink:label="l2" xlink:href="S2.labels.xml"
           xlink:role="http://www.w3.org/1999/xhtml"></span>
10   <!-- Content Base -->
11   <span xlink:type="arc" xlink:from="content" xlink:to="units" xlink:show="none"
           xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"></span>
12   <span xlink:type="locator" xlink:label="units" xlink:href="basic_units.xml"></span>
13   <span xlink:type="locator" xlink:label="content" xlink:href="content.text.html"
           xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
14   <span xlink:type="locator" xlink:label="content" xlink:href="content.img.html"
           xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod"></span>
15 </div>
```

Listing 4.9.: *XLink definition of a multistructured example document*

Formally, the link represents a collection of third-party arcs or more specifically a linkbase, and is the starting point for the document processing.

An example that shows a XLink specification of a multistructured document with two structures is given in Listing 4.9 above. The example embeds the link in a `div` element and its specific definitions in subordinated `span` elements, where the first document structure is defined in lines 2–5, the second document structure in lines 6–9, and the content base in lines 10–14. Both document structures' arcs defined in line 3 and line 7 add their ending resources as linkbases to their starting resources, where the starting resources defined in line 4 and line 8 point to the root nodes of the described structures and the ending resources defined in line 5 and line 9 reference to the according sets of XHTML labels (`role="http://www.w3.org/1999/xhtml"`). Accordingly, the set of basic units is added to the content base's realization sets via linkbase reference (see lines 10–14).

```
<ul id="basic-units">
  <li id="u1" xlink:type="extended">
    <!-- one stone -->
    <span xlink:type="arc" xlink:show="replace"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
    <span xlink:type="locator" xlink:label="_" xlink:href="Birds.content.en.html#en1"
          xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"></span>
    <span xlink:type="locator" xlink:label="_" xlink:href="Birds.content.ja.html#ja1"
          xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"></span>
  </li>
  <li id="u2" xlink:type="extended">
    <!-- two birds -->
    <span xlink:type="arc" xlink:show="replace"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
    <span xlink:type="locator" xlink:label="_" xlink:href="Birds.content.en.html#en2"
          xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"></span>
    <span xlink:type="locator" xlink:label="_" xlink:href="Birds.content.ja.html#ja2"
          xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"></span>
  </li>
</ul>
```

Listing 4.10.: *Basic units of the example document (`Birds.basic_units.html`)*

4.1.6. An Example

This section shows an example description of the already introduced idiom "killing two birds with one stone" from the formal example of Section 3.1.5 as a XLink-based multistructured XML document. I.e. the document is described by means of a content base that consists of basic units and its alternative realizations (CJK characters and roman letters), two document structures that describe a Japanese and an English content structuring, and a multistructured document description that relates all the components of the document to each other.

Content Base

The content base defines the abstract basic units and their specific realizations by means of XLink links. Basic units, Japanese realizations, and English realizations are described within XHTML lists and saved in their own files, each. Listing 4.10 above shows two basic unit definitions that are distinguished by the unique identifiers "`u1`" and "`u2`", where

```
8  <ul>
9    <li id="ja1">&#19968;&#30707;</li><!-- isseki -->
10   <li id="ja2">&#20108;&#40165;</li><!-- nichou -->
11 </ul>
```

Listing 4.11.: *Japanese content nodes of the example document (`Birds.content.ja.html`)*

```
8  <ul>
9    <li id="en1">one stone</li>
10   <li id="en2">two birds</li>
11 </ul>
```

Listing 4.12.: *English content nodes of the example document (`Birds.content.en.html`)*

```
1  <p lang="ja">&#19968;&#30707;&#20108;&#40165;</p>
```

Listing 4.13.: *Japanese representation of the example document*

```
1  <p lang="en">two birds one stone</p>
```

Listing 4.14.: *English representation of the example document*

`u1` represents the content "1 STONE" and `u2` the content "2 BIRDS". The Japanese realizations of the basic units are annotated with the "`http://www.iana.org/assignments/language-subtag-registry#ja`" IRI, the English realizations with "`http://www.iana.org/assignments/language-subtag-registry#en`". These referenced content realizations are distinguished by unique identifiers, again: Listing 4.11 above shows the Japanese content fragments "`ja1`" and "`ja2`" that specify the realizations "一石" and "二鳥"; Listing 4.12 above shows the English fragments "`en1`" and "`en2`" that specify "one stone" and "two birds". The full XHTML files that describe the content base are listed in Appendix B.1.1 on pages 145–146.

```
<ul id="struct">
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
    <span xlink:type="resource" xlink:label="parent" id="node_1"></span>
    <span xlink:type="locator"  xlink:label="child" id="content_1.1"
          xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"
          xlink:href="Birds.basic_units.html#u1"></span>
    <span xlink:type="locator"  xlink:label="child" id="content_1.2"
          xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"
          xlink:href="Birds.basic_units.html#u2"></span>
  </li>
</ul>
```

Listing 4.15.: *Japanese document structure of the example document (`Birds.ja.struct.html`)*

```
<ul id="struct">
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
    <span xlink:type="resource" xlink:label="parent" id="node_1"></span>
    <span xlink:type="locator"  xlink:label="child" id="content_1.1"
          xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"
          xlink:href="Birds.basic_units.html#u2"></span>
    <span xlink:type="locator"  xlink:label="child" id="content_1.2"
          xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"
          xlink:href="Birds.basic_units.html#u1"></span>
  </li>
</ul>
```

Listing 4.16.: *English document structure of the example document (`Birds.en.struct.html`)*

Document Structuring and Labeling

The example document defines two document structures that describe a Japanese and an English output organization of the basic units of the content base. These structures form HyperText Markup Language (HTML) fragments that are described via XLink references. Comparable to the content base, the document structures and labels are defined within XHTML lists again, and saved in their own files, each. Listings 4.13 and 4.14 on the preceding page show the intended inline markup representations of the Japanese and English output structures. Listings 4.15 and 4.16 above show the XLink-based standoff representation of the

```
<ol xml:base="Birds.ja.struct.html" id="labels">
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
    <span xlink:type="locator" xlink:role="http://www.w3.org/2001/XMLSchema#element"
          xlink:label="elem" xlink:href="#node_1"></span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#QName"
          xlink:label="type">html:p</span>
  </li>
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
    <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
    <span xlink:type="locator" xlink:role="http://www.w3.org/2001/XMLSchema#attribute"
          xlink:label="attr" xlink:href="#node_1"></span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#QName"
          xlink:label="type">xmlns:html</span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#string"
          xlink:label="value">http://www.w3.org/1999/xhtml</span>
  </li>
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
    <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
    <span xlink:type="locator" xlink:role="http://www.w3.org/2001/XMLSchema#attribute"
          xlink:label="attr" xlink:href="#node_1"></span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#QName"
          xlink:label="type">html:lang</span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#string"
          xlink:label="value">ja</span>
  </li>
</ol>
```

Listing 4.17.: *Japanese labels of the example document (`Birds.ja.labels.html`)*

Japanese and English output structures, where the output roots' resource descriptions are annotated with the unique identifier "`node_1`" and the content nodes with "`content_1.1`" and "`content_1.2`". Note, that the two structures differ in the order of the referenced basic units and their according language annotations. The output structures' labeling is described via XLink linkbases, where Listing 4.17 above shows the XHTML labeling for the Japanese representation ("`p`" element with a HTML namespace declaration and a "`lang="ja"`" attribute for the root node `node_1`), and Listing 4.18 on the next page shows the XHTML labeling for the

```
<ol xml:base="Birds.en.struct.html" id="labels">
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
    <span xlink:type="locator" xlink:role="http://www.w3.org/2001/XMLSchema#element"
          xlink:label="elem" xlink:href="#node_1"></span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#QName"
          xlink:label="type">html:p</span>
  </li>
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
    <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
    <span xlink:type="locator" xlink:role="http://www.w3.org/2001/XMLSchema#attribute"
          xlink:label="attr" xlink:href="#node_1"></span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#QName"
          xlink:label="type">xmlns:html</span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#string"
          xlink:label="value">http://www.w3.org/1999/xhtml</span>
  </li>
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
    <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
    <span xlink:type="locator" xlink:role="http://www.w3.org/2001/XMLSchema#attribute"
          xlink:label="attr" xlink:href="#node_1"></span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#QName"
          xlink:label="type">html:lang</span>
    <span xlink:type="resource" xlink:role="http://www.w3.org/2001/XMLSchema#string"
          xlink:label="value">en</span>
  </li>
</ol>
```

Listing 4.18.: *English labels of the example document (`Birds.en.labels.html`)*

English representation (“p” element with a HTML namespace declaration and a “`lang="en"`” attribute for the root node `node_1`). The full XHTML files that describe the Japanese and English document’s standoff structuring and labeling are listed in Appendix B.1.1 on the pages 147 ff.

```
8  <div xlink:type="extended">
9    <!-- Structure: Japanese output -->
10   <span xlink:type="arc" xlink:from="struct.ja" xlink:to="labels.ja" xlink:show="embed"
           xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"></span>
11   <span xlink:type="locator" xlink:label="struct.ja"
           xlink:href="Birds.ja.struct.html#node_1"></span>
12   <span xlink:type="locator" xlink:label="labels.ja"
           xlink:role="http://www.w3.org/1999/xhtml"
           xlink:href="Birds.ja.labels.html#labels"></span>
13   <!-- Structure: English output -->
14   <span xlink:type="arc" xlink:from="struct.en" xlink:to="labels.en" xlink:show="embed"
           xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"></span>
15   <span xlink:type="locator" xlink:label="struct.en"
           xlink:href="Birds.en.struct.html#node_1"></span>
16   <span xlink:type="locator" xlink:label="labels.en"
           xlink:role="http://www.w3.org/1999/xhtml"
           xlink:href="Birds.en.labels.html#labels"></span>
17   <!-- Content Base -->
18   <span xlink:type="arc" xlink:from="content" xlink:to="units" xlink:show="none"
           xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"></span>
19   <span xlink:type="locator" xlink:label="units"
           xlink:href="Birds.basic_units.html#basic-units"></span>
20   <span xlink:type="locator" xlink:label="content" xlink:href="Birds.content.en.html"
           xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"></span>
21   <span xlink:type="locator" xlink:label="content" xlink:href="Birds.content.ja.html"
           xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"></span>
22 </div>
```

Listing 4.19.: *Multistructured document definition of the example document (`Birds.doc.out.html`)*

Multistructured Document

The multistructured document description puts all files of the content base and the document structures in relation to each other via XLink. The list of basic units is defined as linkbase for the lists of language-specific content realizations and the output labels as linkbase for the output structures. Listing 4.19 above shows the multistructured document description as a single XLink reference that is embedded in a XHTML `div` element; Appendix B.1.1 on page 150 lists the document's full XHTML file.

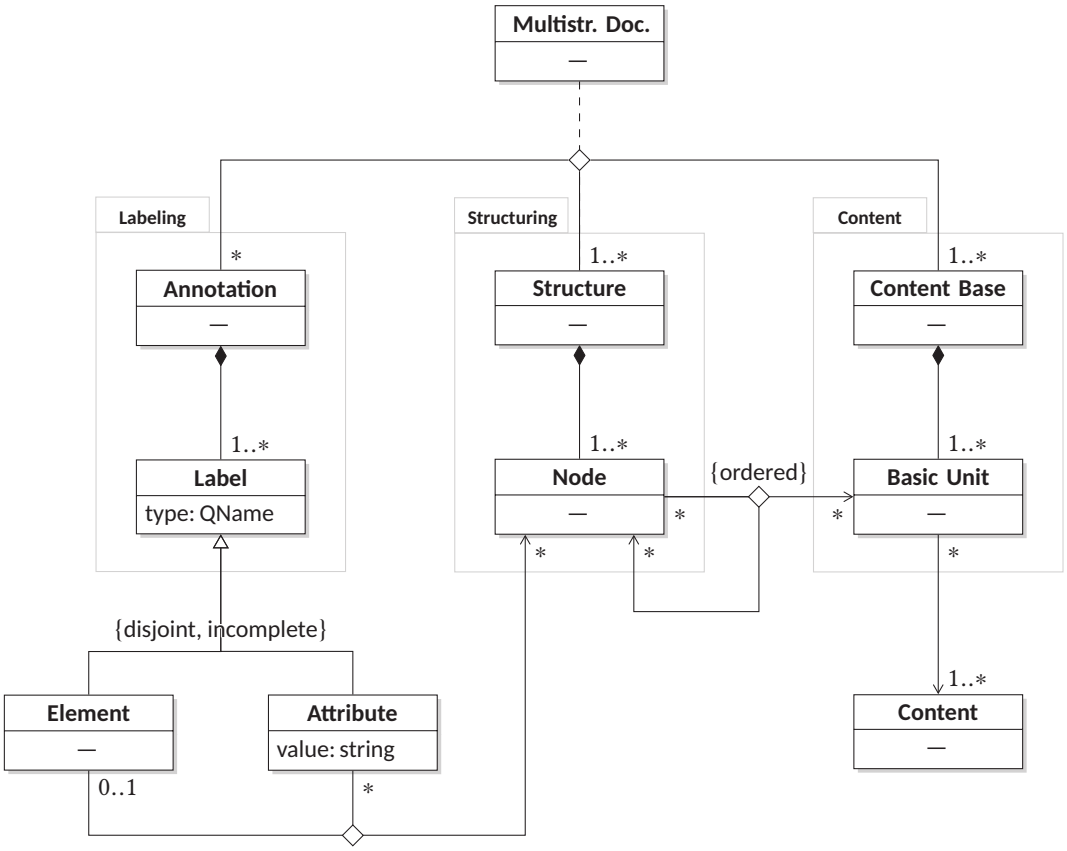


Figure 4.3.: *Schematic data model of multistructured documents and their components*

4.1.7. Discussion

The previous sections have introduced a generic description approach for multistructured documents based on established standards of the XML ecosystem. By using XLink as container language this approach can be incorporated in different host languages and processing frameworks—the preceding example used XHTML as host language. As the document structures' labels are generically described by means of XML Schema expressions, the specific document structures can represent any XML-based document type according to its original specification.

Figure 4.3 above summarizes the underlying data model that is specified by the previously introduced generic description language for multistructured documents. This model is described by means of an Unified Modeling Language (UML) class diagram, based on the

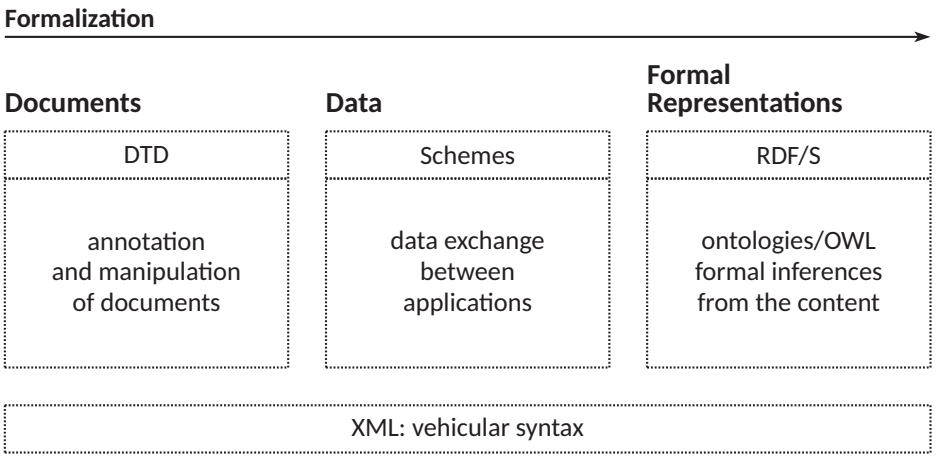


Figure 4.4.: *XML and the "grammatization" of documents (cf. Pédauque 2006c, Fig. 1, translated by F. S.)*

XML modeling techniques provided and introduced by Eckstein and Eckstein (2004) that give insight into the relationships between the document's components. The diagram shows a multistructured document as a ternary association between the document's labeling, structuring, and content. A document's labeling is composed of annotation sets that describe specific, distinguishable element and attribute labels (amongst others). These labels are associated to components of a structure. A document's structure is a composition of nodes that link to each other and to components of a content base. A document's content base is a composition of basic units that are associated with specific contents. These contents are distinguishable into realization groups of specific content modes, e.g. a text, table, or image mode. This differentiation is not shown in the figure for brevity. Furthermore it is implied, that an attribute label is only associated with nodes that are associated with an element label as well and that the reflexive association for nodes does not lead to cyclic structures.

The introduced description framework incorporates and reconciles different conceptual document views with different degrees of document formalization and creates possibilities that even may exceed the intended use for multichannel publishing. Figure 4.4 above shows the different document approaches that are covered by different XML standardization efforts and that are all incorporated within the introduced model for multistructured documents. The least formal and most comprehensible document description approach via Document Type Definitions (DTDs) is used in our framework in the context of distinguishable content realizations as they are provided to a human reader (see Tab. 4.4, p. 48). The next more formal approach that introduces the concept of data types to XML via XML Schema is used in our framework to describe explicitly the document's labeling (see Tab. 4.3, p. 47). The

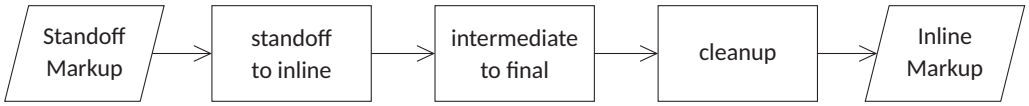


Figure 4.5.: *Standoff to inline markup transformation*

last and most formal approach that describes information as a graph of resources via RDF is used in our framework to relate all components in a document to each other (see Tab. 4.2, p. 46).

As the multistructured document's structures are described via embedded XML Schema annotations, its export to XML Schema Definitions (XSDs) might support the generalization of document instances, i.e. the definition of document types. In parallel, the export of the embedded RDF statements might open the processing of multistructured documents to semantic web technologies. The exploitation of RDF that is embedded in XLink has been discussed e.g. by Daniel (2000). Alternatively, a further formalization and more generic approach respectively might be to describe a document structure and its labels by means of the XML Information Set as described by Tobin (2001). While these ideas might lead to technical processing environments that are uncommon to actual publishing systems they might be beneficial to future work. The processing of multistructured documents as introduced and its use within multichannel publishing workflows is described in the following Sections 4.2 and 4.3.

4.2. Multistructured Document Processing

The previously introduced description framework provides a declarative approach to define multistructured documents by means of common XML standards. The problem of processing the previously described DAGs with common XML tools that expect clean tree structures is addressed in this section.

To keep the introduced framework interoperable with common publishing tools and systems, the traversal paths described via XLink have to be transformed to tree structures for each of the DAG's sources. In other words: the standoff document description has to be transformed to corresponding sets of inline representations in order to be processable by common XML tools. Figure 4.5 above conceptualizes a processing pipeline that translates the XLink-based

standoff markup described in the preceding Section 4.1 to regular inline markup. The specific implementation of the pipeline's sub-steps is addressed in the following sections by means of consecutive XSLT steps (Kay 2007) that are arranged by a XML Pipeline Language (XProc) instance (Thompson et al. 2010).

As preliminary work, Section 4.2.1 describes a XSLT stylesheet that provides general XLink traversal functionality. This is used in the XSLT implementation of the standoff to inline markup transformation described in Section 4.2.2. This transformation stylesheet traverses and collects all components of a XLink-based document description and turns it into an intermediate inline representation. Sections 4.2.3 and 4.2.4 describe how the intermediate representation of the inline markup is transformed to its final representation. Finally, Sections 4.2.5 and 4.2.6 conclude the introduction of the standoff to inline markup transformation pipeline with an example and a discussion.

4.2.1. XSLT-based XLink Traversal

As a XSLT processor does not interpret XLink descriptions on its own, the traversal behavior of given links has to be defined, first. Therefore, the general XLink processing is implemented in its own XSLT stylesheet that copies a given document, but recognizes and replaces embedded XLink annotations by traversing the given links and including the addressed resources. The traversal of extended links is processed as described by the XLink specification (Maler et al. 2010), but with additional processing behavior for cases, where addressed remote resources are themselves again part of another XLink description again, as this is not specifically anticipated by the standard. Figure 4.6 on the next page shows a simplified structogram (DIN 66261) of the extended link processing and Appendixes C.1.1 and C.1.2 its XSLT implementation that are jointly explained in the following.

The link processing is initiated by the template that matches XLink extended-type elements ("`*[@xlink:type='extended']`", see Appx. C.1.1, ll. 24–59). As the XLink standard allows to describe link arcs implicitly, all arcs have to be made explicit in order to be matchable in the further XSLT processing (see Appx. C.1.1, ll. 212–236). This means that missing `from`- and `to`-attributes are added to XLink arc-type elements to assemble all the resources' labels of the link to address the according starting- and ending-resources (see Appx. C.1.1, ll. 238–272). Furthermore, each arc's `from`- and `to`-label pair is translated into its own arc-type element (see Appx. C.1.1, ll. 274–300). These explicit arcs are handed over to matching

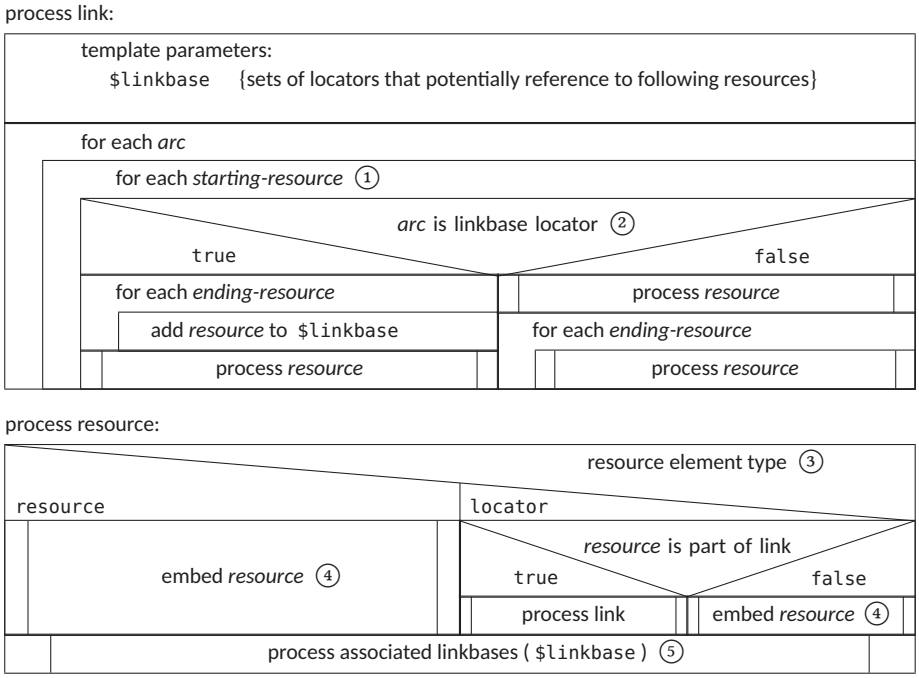


Figure 4.6.: *Embedded XLink processing*

templates, where the link is further processed in the order of addressed starting-resources ("`*[@xlink:type='arc']`", see Appx. C.1.1, ll. 62–83 and Fig. 4.6 at No. ①) and depending on whether the arc is specifying a linkbase location or a regular link (see Appx. C.1.1, ll. 86–131 and Fig. 4.6 at No. ②). In the case of a linkbase locator the addressed linkbases are made accessible to further resource processing and only the starting-resource is handed over to a matching template. In the case of a regular link the starting-resource is processed by applying templates first followed by all corresponding ending-resources.

The resource processing is specified in a template that is triggered by XLink resource- and locator-type elements ("`*[@xlink:type='resource' or @xlink:type='locator']`", see Appx. C.1.1, ll. 136–185 and Fig. 4.6 at No. ③). Local resources and remote resources that are not part of another XLink definition again are handed over to an identity template (see Appx. C.1.1, ll. 190–197) that copies the given XML structure and processes eventually embedded extended links as described before (see Appx. C.1.1, ll. 200–207 and Fig. 4.6 at No. ④). Remote resources that are part of another link description lead to a processing of this new link as previously explained, but with special treatment depending on whether the originally addressed resource is an arc-, locator-, or resource-type element (calling these

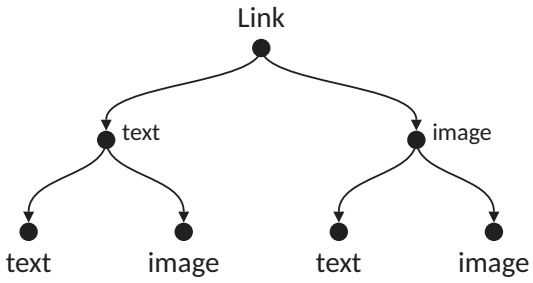


Figure 4.7.: *Graphical representation of the processed example link defined in Listing 4.3*

elements "focused"). If the addressed element is an arc-type element, then only this specific arc is used during link traversal. If the addressed element is a locator- or resource-type element, then only arcs with this specific starting-resource are traversed. Any of these resource processing cases are followed by retrieving and processing the links from accessible linkbases that address the actual resource as starting-resource (see Appx. C.1.1, ll. 175–181 and Fig. 4.6 at No. ⑤).

Some of the previously described design and implementation choices for the XLink processing have a significant impact on the output order of the given resources. While the order of XLink annotations does not affect the described graph (i.e. different descriptions can define the very same graph), its recursive traversal processing automatically determines an output order, as XSLT interprets all processed XML in- and output in terms of ordered tree structures. Furthermore, the XLink specification does not define how to embed XML-based ending-resources into respective starting-resources (see Maler et al. 2010, Sec. 5.6.1). Rather than choosing which XML node of the starting resource to use to embed its ending-resources, starting- and ending-resources are issued successively. In the given XLink processing a link arc is issued for a given starting-resource as follows: first the starting-resource itself succeeded by all associated ending-resources that are given by accessible linkbases, then all ending-resources as associated by the original arc that are each again succeeded by their associated linkbase ending-resources. As each arc is processed only once within a link, as a result, the sequential link processing even copes with cyclic arc definitions by transforming them to regular XML trees. E.g. the cyclic arc definition of the basic unit shown in Listing 4.3 and visualized in Figure 4.1 on page 49 connects all associated resources of the link with one another. Specifically, following the annotation order in Listing 4.3, first the text-node is linked to itself and the image-node, then the image-node is linked back to the text-node and to itself. Figure 4.7 above shows the tree structure that would result from the XSLT

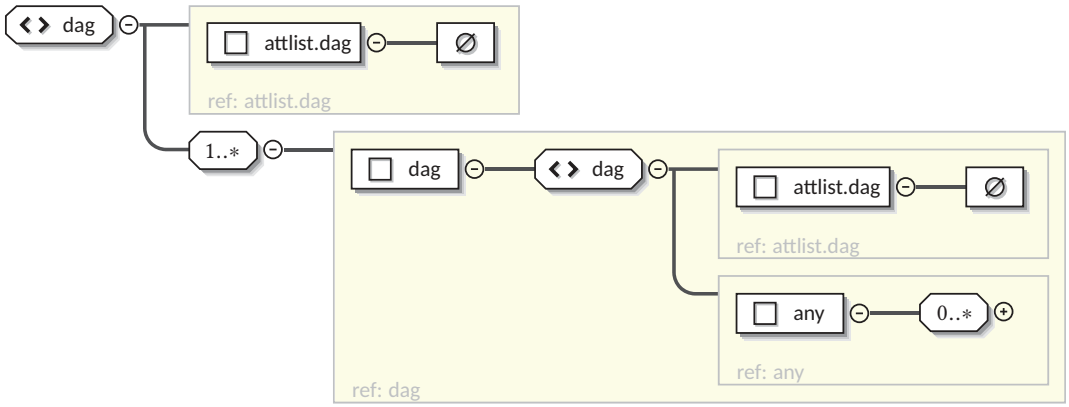


Figure 4.8.: *Document structure schema*

processing of this arc. Note that in the case at hand, where all existing resources are linked to one another, the processing leads to a multiplication of the output ending resources that is equal to the number of starting resources raised to the power of 2.

4.2.2. Standoff to Inline Markup Transformation

The first step of the standoff to inline markup transformation pipeline (see Fig. 4.5, p. 63) is described in this section. This step uses the XLink processing introduced in the preceding Section 4.2.1, but translates the specific link annotations for multistructured documents (see Section 4.1) to pipeline specific markup. This step is implemented as a XSLT stylesheet that imports and extends the XLink traversal stylesheet from Appendix C.1.1 by taking special resource- and arc-roles into account and transforming their standoff-specific annotations into an intermediate inline representation. Appendix C.1.3 lists the XSLT implementation. Its processing is described in more detail in the following.

The initial processing step of the stylesheet, i.e. the template that matches the document root (see Appx. C.1.3, lines 19–41), assumes that the given document contains XLink descriptions for multistructured documents that refer in particular to one or more structure roots and associate these structures with their corresponding labeling (see Section 4.1.5). Each of these document structures is wrapped in a `dag:dag`-element and assembled in a common superordinated `dag:dag`-element again. Figure 4.8 above shows the resulting intermediate XML schema of this superstructure for multistructured documents.

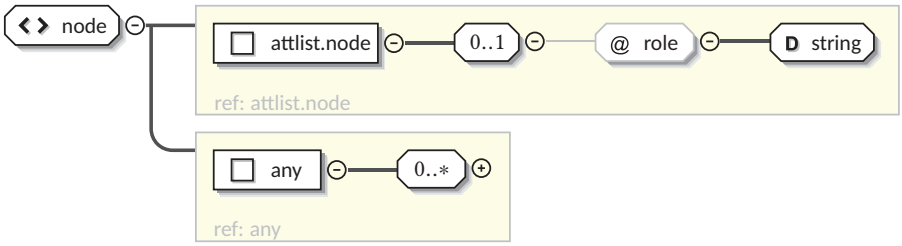


Figure 4.9.: *Structure node schema*

Figure 4.10.: *Element label schema*

The default behavior for all following ending resources in the process of recursive link traversal is to wrap them in `dag:node`-elements together with their according role annotations that specify e.g. content mode choices (see Appx. C.1.3, lines 166–182). This applies in particular to the nested links that associate each parent node of a document structure to its children (see Section 4.1.3). This ensures, that the graph of links is transformed to a processable XML tree and that all following labeling annotations are joint together within their according structure node. Figure 4.9 above shows the intermediate XML schema for structure nodes.

The labeling of the document structures is described by links with special resource roles (see Section 4.1.4). The stylesheet differentiates element and attribute labels, where element labels are transformed to empty `dag:element`-elements with a `type`-attribute that specifies the element type by means of a qualified XML name (see Appx. C.1.3, lines 83–117). As the attribute labeling consists of an attribute type and an attribute value that are described by their own link arcs each, the stylesheet creates for both cases its own empty `dag:attribute`-elements (see Appx. C.1.3, lines 120–161) that are associated to each other by an `id`-attribute. In the case of an attribute type description the element gets a `type`-attribute that specifies the attribute type by means of a qualified XML name; in the case of an attribute value description the element gets a `value`-attribute that specifies the attribute value. Element and attribute labels are associated to each other implicitly by a common parent structure

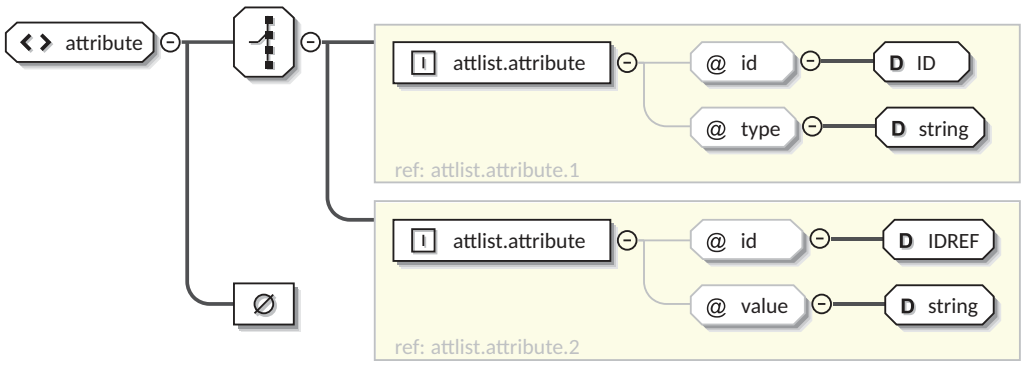


Figure 4.11.: *Attribute label schema*

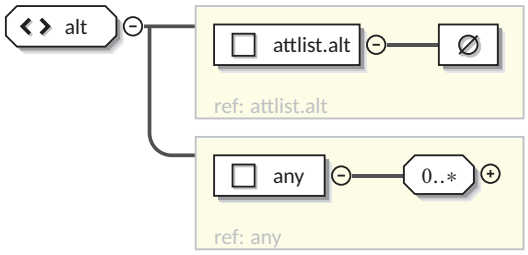


Figure 4.12.: *Basic unit schema*

node. Figure 4.10 on the preceding page shows the intermediate XML schema for element labels and Figure 4.11 above for attribute labels.

Alternative resources, i.e. resources that are associated to each other by an arc with a respective role annotation as e.g. used for the description of basic units with their alternative content nodes (see Section 4.1.2), are wrapped by a `dag:alt`-element (see Appx. C.1.3, lines 46–80). As described previously at the end of Section 4.2.1 on page 66, the automated traversal of link arcs that describe cyclic graphs, as it is the case for basic units, leads to a multiplication of each resource alternative by the total number of alternative resources associated within the arc. To prevent this behavior during the transformation of cyclic arcs to tree structures and ensure that each alternative resource is output only once, just one of the alternative resources is used as a starting resource (see Appx. C.1.3, lines 65–69). Figure 4.12 above shows the intermediate XML schema for alternative nodes.

4.2.3. Intermediate to Final Markup Transformation

With the previously described processing step (see Section 4.2.2), the standoff markup representation of a multistructured document is transformed to an inline markup representation that uses intermediate XML elements to distinguish alternative document structures (`dag:dag`), structure nodes (`dag:node`), structure labels (`dag:element`, `dag:attribute`), and alternative contents (`dag:alt`). Next this intermediate representation is transformed to its final intended inline representation by a subsequent XSLT step as described in the following.

The XSLT stylesheet (see Appx. C.1.4) copies the given intermediate document, but provides for each intermediate annotation case its own template that specifies the transformation of the intermediate XML description to the intended final XML structure. The transformation of element node descriptions to actual XML elements is implemented in the template matching those structure nodes that contain an element label ("`dag:*[dag:element]`", see Appx. C.1.4, lines 31–47). As the type of XML elements is unambiguous, only one element label per structure node is allowed. In the case of a successful XML element creation (only one element label exists), first all attribute labels that are associated with the currently processed structure node are processed and added to the newly created element, followed by all further subordinated nodes of the current structure node being processed as the new element's child nodes. The transformation of attribute node descriptions to actual XML attributes is implemented in the template matching those attribute labels that describe the attribute type ("`dag:attribute[@type]`", see Appx. C.1.4, lines 50–67). The attribute type label's corresponding attribute value labels are collected by means of their unique identifier references and jointly transformed to a XML attribute node. The transformation of alternative content node descriptions is implemented in the template that matches the according intermediate annotation ("`dag:alt`", see Appx. C.1.4, lines 88–119). Depending on the content mode choice that is passed as a template parameter, only one of the alternative content realizations is issued in the final document creation. The template throws warning messages in the case of results where e.g. no or more than one content realization matches the given content mode.

4.2.4. Final Markup Cleanup

In terms of XML data objects, the preceding transformation steps provide the final inline representations of the particular document structures from the former standoff markup. The last processing step of the entire standoff to inline markup transformation pipeline (see Fig. 4.5 on p. 63) provides merely "aesthetic" markup changes: the created document structures are again processed by a further XSLT stylesheet (see Appx. C.1.5) that reorganizes the XML namespace markup by taking the namespace declarations' scope into account (see Thompson et al. 2009, Sec. 6).

4.2.5. An Example

With the standoff to inline markup transformation pipeline fully described, this section revisits the standoff document description of the idiom "killing two birds with one stone" from Section 4.1.6 and follows its consecutive transformations from the original standoff description to its target inline representation. The source document of the transformation pipeline is the multistructured document shown in Listing 4.19 on page 60 that puts the idiom's Japanese and English content realizations in relation to their respective output structures.

Intermediate Markup Representation

The first transformation step traverses and collects all components of a multistructured document and translates it with its XLink annotations to a pipeline specific intermediate inline representation. Listing 4.20 on the next page shows the intermediate inline representation that is created from the idiom's source document. The intermediate representation of the Japanese document structure is listed in lines 2–20, where line 3 denotes the structure root's element label as of type "`html:p`", lines 4 and 5 the element's namespace declaration by means of an attribute label of type "`xmlns:html`" and value "`http://www.w3.org/1999/xhtml`", and lines 6 and 7 an ordinary attribute label of type "`html:lang`" and value "`ja`". Lines 8–13 and 14–19 show each a structure node description that sets the content modes to Japanese realizations by means of a `role` attribute. Both structure nodes contain alternative content

```
<dag:dag xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags">
  <dag:dag>
    <dag:element type="html:p"/>
    <dag:attribute id="d6e22" type="xmlns:html"/>
    <dag:attribute id="d6e22" value="http://www.w3.org/1999/xhtml"/>
    <dag:attribute id="d6e36" type="html:lang"/>
    <dag:attribute id="d6e36" value="ja"/>
    <dag:node role="http://www.iana.org/assignments/language-subtag-registry#ja">
      <dag:alt>
        <dag:node role="http://www.iana.org/assignments/language-subtag-registry#ja">
          &#19968;&#30707;</dag:node>
        <dag:node role="http://www.iana.org/assignments/language-subtag-registry#en">
          one stone</dag:node> ↲ </dag:alt> ↲ </dag:node>
    <dag:node role="http://www.iana.org/assignments/language-subtag-registry#ja">
      <dag:alt>
        <dag:node role="http://www.iana.org/assignments/language-subtag-registry#ja">
          &#20108;&#40165;</dag:node>
        <dag:node role="http://www.iana.org/assignments/language-subtag-registry#en">
          two birds</dag:node> ↲ </dag:alt> ↲ </dag:node>
  </dag:dag>
  <dag:dag>
    <dag:element type="html:p"/>
    <dag:attribute id="d25e22" type="xmlns:html"/>
    <dag:attribute id="d25e22" value="http://www.w3.org/1999/xhtml"/>
    <dag:attribute id="d25e36" type="html:lang"/>
    <dag:attribute id="d25e36" value="en"/>
    <dag:node role="http://www.iana.org/assignments/language-subtag-registry#en">
      <dag:alt>
        <dag:node role="http://www.iana.org/assignments/language-subtag-registry#ja">
          &#20108;&#40165;</dag:node>
        <dag:node role="http://www.iana.org/assignments/language-subtag-registry#en">
          two birds</dag:node> ↲ </dag:alt> ↲ </dag:node>
    <dag:node role="http://www.iana.org/assignments/language-subtag-registry#en">
      <dag:alt>
        <dag:node role="http://www.iana.org/assignments/language-subtag-registry#ja">
          &#19968;&#30707;</dag:node>
        <dag:node role="http://www.iana.org/assignments/language-subtag-registry#en">
          one stone</dag:node> ↲ </dag:alt> ↲ </dag:node>
  </dag:dag>
</dag:dag>
```

Listing 4.20.: *Intermediate representation of the example document*

```
1 <html:p xmlns:html="http://www.w3.org/1999/xhtml" lang="ja">&#19968;&#30707;
   &#20108;&#40165;</html:p>
```

Listing 4.21.: *Inline output of the Japanese example structure*

```
1 <html:p xmlns:html="http://www.w3.org/1999/xhtml" lang="en">two birds one stone</html:p>
```

Listing 4.22.: *Inline output of the English example structure*

```
1 <p xmlns="http://www.w3.org/1999/xhtml" lang="ja">&#19968;&#30707;&#20108;&#40165;</p>
```

Listing 4.23.: *Final output of the Japanese example structure*

```
1 <p xmlns="http://www.w3.org/1999/xhtml" lang="en">two birds one stone</p>
```

Listing 4.24.: *Final output of the English example structure*

realizations with according mode annotations that are described via `role` attributes and from which the final content is chosen in the following processing. Accordingly, lines 21–39 show the intermediate representation of the English document structure.

Inline Markup Representation

The second transformation step translates the intermediate representation of the multistructured document to its intended inline target representations. Listing 4.21 above shows the created target representation of the Japanese and Listing 4.22 above of the English document structure with their reconstructed XML elements, namespace declarations, and attributes, where the Japanese structure contains the CJK character and the English structure the roman letter realizations of the content.

Final Markup Representation

The last transformation step reorganizes the markup of the created XML documents by abbreviating the namespace declarations and their according qualified name prefixes in order to render the markup more human readable. Listings 4.23 and 4.24 on the preceding page show the final output from the processing pipeline, where the "`html`" prefix of the "`http://www.w3.org/1999/xhtml`" namespace could be eliminated.

4.2.6. Discussion

The previous sections have introduced a processing pipeline to transform XLink-based standoff document descriptions as introduced in preceding Section 4.1 to regular inline descriptions as expected by common XML processing tools. As the introduced pipeline is implemented via XSLT stylesheets, the standoff markup processing does not rely on additional software, but integrates itself to established publishing systems by means of basic XML processing.

The UML diagram in Figure 4.13 on the next page gives an overview of the implementation architecture with its components and dependencies. The whole processing pipeline is assembled and packaged via XProc (Thompson et al. 2010) as shown in Appendix C.1.6 that uses consecutively the standoff to inline, intermediate to final, and cleanup stylesheets to transform documents that are described via standoff markup to their inline markup representations. Basic transformation tasks as the XLink traversal implementation and the XML Path Language (XPath) function library for DAG are provided and imported as independent stylesheets and thereby reusable in further application scenarios.

As the implemented XSLT stylesheets cope with both standoff and inline markup, the introduced processing pipeline does not only work for the standoff markup description framework established in Section 4.1, but also for less generic description approaches. E.g. the deconstruction of document structures and the separation of the document labeling from the structuring might be undesired in cases, where specific document structures are preferred to be edited directly within their intended inline representations without any pre-processing. Other frameworks (e.g. Chatti et al. 2007) follow this approach by using common inline descriptions for the document structures and linking the structures' leafs to

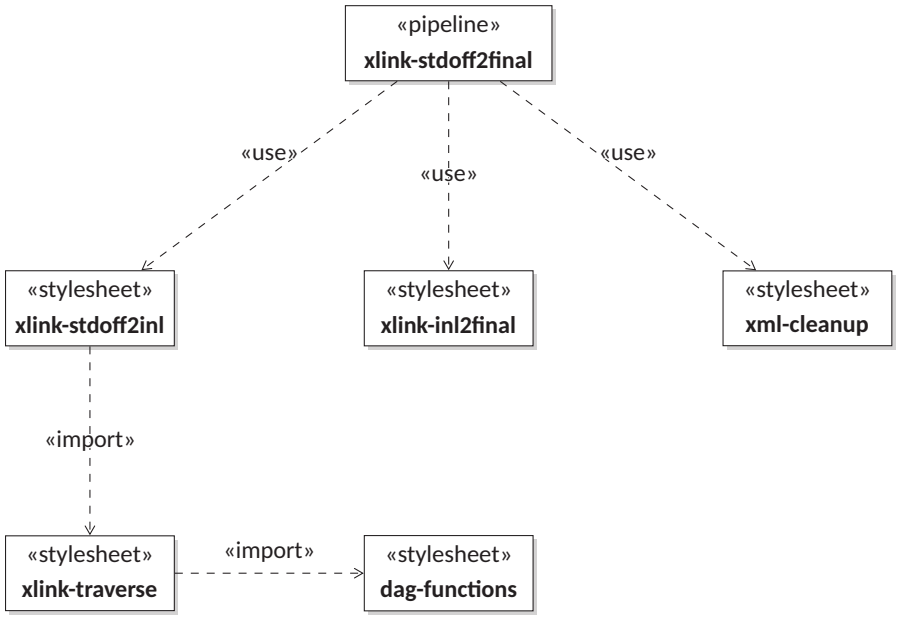


Figure 4.13.: *Schematic diagram of the multistructured document processing components*

corresponding disjoint content fragments. Accordingly, the introduced processing pipeline copes with common inline XML structures that embed XLink references to basic units as described in Section 4.1.2 and produces the intended inline representations. This is possible since the whole processing framework is not depending on specific XML document types, but on XLink attributes only.

Furthermore, as the standoff markup semantics are described and processed by means of link roles, the introduced processing pipeline is not depending on a specific XLink description syntax: indeed, Section 4.1 introduced specific XLink descriptions for document structuring, labeling, and content, but as long as the agreed link annotations are used via according IRIs, the choice whether a standoff descriptions is specified e.g. by means of inbound or outbound links is irrelevant to the processing pipeline and transformed either way. That applies to the addressing syntax as well: as long as the used XSLT processor provides according functionality, it is irrelevant to the processing pipeline whether fragments are referenced e.g. via fragment identifiers or XPointer.

The description and processing of multistructured documents as introduced so far, provides a technical foundation to introduce the genre-aware document model of Chapter 3 into multichannel publishing workflows. This specific application is described in the following Section 4.3.

4.3. Genre Definition and Processing

This section describes how the previously introduced document description and processing framework for multistructured documents can be used for a technical implementation of the alternative document perspectives introduced by the model of genre in Chapter 3 and how to take advantage of it in the context of multichannel publication workflows. Therefore, the three document perspectives of form, content, and medium can be described by distinct document structures within a common multistructured document. For each of these perspectives predefined XML markup languages exist: e.g. conventional form structures (vu) can be described by distinct XHTML modules (Austin et al. 2010 and Tab. 4.4 on p. 48), the content structures (lu) by means of the RST format (O'Donnell 2000), and the medium structures (su) by a navigation format as e.g. the Navigation Control file for XML applications (NCX) (ANSI/NISO Z39.86-2005 (R2012)).

As explained in Chapter 3, the different document perspectives can be described by means of their relations to each other. More formally: patterns of matching element configurations between the annotation trees constitute genres; if e.g. a specific RST structure (lu) generally comes in conjunction with a specific output realization (vu), this combination can be considered as typical for a specific group of documents. These relations can be described by means of XSLT stylesheets (Kay 2007) that specify these relations as transformation rules between the different document perspectives' markup languages. In the context of document creation this is in particular a transformation from a given content structure (RST) to specific form (XHTML) and medium (NCX) realizations. More formally: the RST description is the primary structure that the genre-defining relation patterns are based on. Such a genre is primarily defined by restrictions and rules for the content structure and secondarily defined by rules of matching configurations between the content and the form and navigation structures. Furthermore, starting from the content structure, there can be alternative rules of matching configurations that compose alternative subgenres that differ e.g. in their used output media.

Figure 4.14 on the next page shows the conceptual implementation of the processing pipeline that transforms a "generic" document that is described by its content structure to its according genre-specific realizations. The specific steps of the pipeline's technical implementation are described in the following sections. Preliminary to the description of the pipeline steps, Section 4.3.1 introduces the generic content structuring by means of RST that is used here as primary structure of a genre-definition. The specification of expected key structures

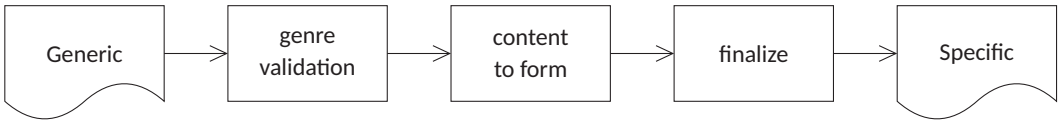


Figure 4.14.: *Generic to genre-specific document transformation*

within the primary structure for a given genre by means of validation rules is described in Section 4.3.2. The specification of dependencies between primary and co-structures that define how a key structure within the primary structure is transformed to its according co-structures for a given subgenre is described in Section 4.3.3, the final markup cleanup of the created co-structures in Section 4.3.4. Finally, Sections 4.3.5 and 4.3.6 conclude the technical genre definition and processing introduction with an example and a discussion.

4.3.1. Generic Content Description

The generic organization of a document's content, i.e. a document organization that is independent of its later representation, is described in the introduced framework by means of the Rhetorical Structure Theory (RST). The RST as originally introduced by Mann and Thompson (1988) is conceptualized to describe the relations among segments of written monologue, but has developed to a description language for content structures in multimodal documents in general (see e.g. Bateman 2008, Chap. 4). The RST describes a document structure by means of nested relation schemas that form a tree with the document's content segments as its leaves. Each of these schemas describes a specific type of relation that can be assigned to one of the two groups of hypotactic or paratactic relations. The first group distinguishes a superordinated from its subordinated content node, the so-called "nucleus" from its "satellite". I.e., there is a main statement (the nucleus) that is supported by a potentially omittable side-statement (the satellite). The second group provides relation patterns that do not subordinate any of its participating nodes and form so-called "multinuclear" relations. Figure 4.15 on the next page shows the schematic representations of nucleus-satellite and multinuclear relations. The individual relation types are distinguished by specific constraints on the contents and combinations of the participating nuclei (*N*) and satellites (*S*) and by the intended effect of the writer (*W*) on the reader (*R*). E.g. Table 4.5 on the next page shows the RST definition of the "evidence" relation as provided by Mann and Thompson (1988). The nucleus-satellite relations can be further divided into two groups of relation types:

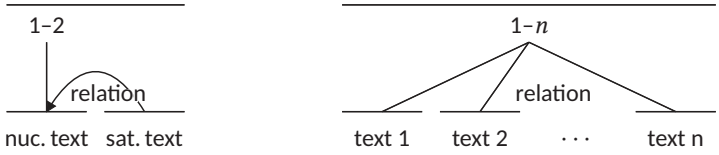


Figure 4.15.: *Schematic representations of (a) nucleus-satellite and (b) multinuclear RST relations*

relation name:	Evidence
constraints on N:	R might not believe N to a degree satisfactory to W
constraints on S:	R believes S or will find it credible
constraints on N + S combination:	R's comprehending S increases R's belief of N
the effect:	R's belief of N is increased

Table 4.5.: *RST definition of the "evidence" relation (Mann and Thompson 1988, p. 251)*

one group describes the subtle embedded rhetorical relations that are used to convince the reader of a main statement without focusing the relation itself. E.g. the evidence relation shown in Table 4.5 above is a member of this group of so-called "presentational" relations. The second group describes the so-called "subject-matter" relations that are used to convey a statement by recognizable rhetorical structures. E.g. the "evaluation" is a member of subject-matter relations. Appendix A.5 provides the full list of presentational, subject-matter, and multinuclear relation definitions that are part of the so-called "classic RST" (see Mann and Taboada 2005).

For the processing of RST structures, O'Donnell (2000) introduces a XML description language along with a graphical user interface to create RST annotations. But, even though the RST uses tree structures by definition, the given RST description language deconstructs the content tree and uses a paratactic description of the structure nodes that are organized by referencing one another. I.e. the described RST structure is not represented analogously in the XML structure. As a result, the explicitly described parent-children relations in the RST tree are not directly interpretable by standard XML processing (e.g. via XPath as used in XSL processing). To overcome this issue, a XSLT stylesheet has been implemented to transform the "flat" RST description to a "deep", nested equivalent that reconciles the XML representation with the described RST structure. To keep this new RST description language compatible with the original RST software, a second XSLT stylesheet describes the back-transformation

```
<group xmlns="http://www.sfu.ca/rst">
  <segment>nuc. text
    <segment relname="relation" reltype="rst">sat. text</segment>
  </segment>
</group>
```

Listing 4.25.: *XML description of the nucleus-satellite RST relation shown in Figure 4.15a*

```
<group xmlns="http://www.sfu.ca/rst">
  <segment relname="relation" reltype="multinuc">text 1</segment>
  <segment relname="relation" reltype="multinuc">text 2</segment>
  <segment relname="relation" reltype="multinuc">text 3</segment>
</group>
```

Listing 4.26.: *XML description of the multinuclear RST relation shown in Figure 4.15b for* $n = 3$

to the flat RST description. The DTD of the original RST format as provided by O'Donnell (2000) is shown in Appendix C.2.1 and its deep reimplementation along with an additional Schematron schema (ISO/IEC 19757-3 2006) in Appendixes C.2.2 and C.2.3. Appendixes C.2.5 and C.2.6 show the XSLT stylesheets for the flat to deep and deep to flat RST transformations, Appendix C.2.7 a stylesheet that visualizes a RST structure by transforming it to a Scalable Vector Graphics (SVG) instance. All graphical RST structure representations in this thesis are SVG renderings of RST to SVG transformations. E.g. Figure 4.15 on the preceding page shows example SVG renderings of the two RST descriptions from Listings 4.25 and 4.26 above. The new "deep" RST description language and its validation are introduced in more detail in the following section.

4.3.2. Genre-specific Content Validation

The first step of the generic to genre-specific document transformation (see Fig. 4.14, p. 77) verifies whether a given content structure is a member of a specific genre or not. While a DTD and a Schematron schema describe the general XML structure of a RST tree (see Appx. C.2.2 and C.2.3), a genre might be defined by allowing only a subset of possible RST schemas and allowing only specific schema combinations. To describe these genre-specific restrictions, a set of XSLT functions was implemented that identify specific RST characteristics of a given RST node by means of XPath expressions, as e.g. whether the node describes specific con-

```
The given structure contains 2 text node(s)
• group → element(/1)
  group is a nucleus
  group has 1 child(ren)
• segment → element(/1/1)
  segment is a nucleus
  segment's text is "nuc. text"
  segment has 1 child(ren)
• segment → element(/1/1/1)
  segment is a satellite
  segment is in "relation" relation to its nucleus
  segment's text is "sat. text"
```

Listing 4.27.: *Validation report for the nucleus-satellite RST relation shown in Listing 4.25*

```
The given structure contains 3 text node(s)
• group → element(/1)
  group is a nucleus
  group has 3 child(ren)
• segment → element(/1/1)
  segment is multinuclear
  segment is in "relation" relation to its co-nuclei
  segment's text is "text 1"
• segment → element(/1/2)
  ...
```

Listing 4.28.: *Validation report for the multinuclear RST relation shown in Listing 4.26*

tent (`self::rst:segment`) or a superordinated structure (`self::rst:group`), whether a node is a nucleus (`not(@reltype)`), a satellite (`@reltype = 'rst'`), or multinuclear (`@reltype = 'multinuc'`), and in which relation a node is to its context (`@relname`).[7] Table 4.6 on the next page summarizes the implemented XSLT functions that can be used to describe the genre-specific content restrictions by extending the general Schematron schema from Appendix C.2.3. Listings 4.27 and 4.28 above show a general Schematron validation report for the two RST descriptions from Listings 4.25 and 4.26 on the preceding page. Appendix C.3.2 shows the XProc implementation of this validation step, that follows a preceding standoff to inline markup transformation. An example definition of a specific genre and its content restrictions follows in Section 4.3.5.

7. Note, that the specific XML attributes and attribute-values are inherited from original RST-DTD (O'Donnell 2000).

Functions	**Summary** (line references Appx. C.2.4)
`contains_text($node)`	Test if a given node contains non-trivial text nodes. (ll. 19–26)
`get_text($node)`	Select non-trivial text nodes. (ll. 29–36)
`has_relation($node,$name)`	Test if a given node is in a given RST relation. (ll. 41–50)
`is_nucleus($node)`	Test if a given node is a RST nucleus (non-multinuclear). (ll. 55–62)
`is_nucleus_chain($first,$last)`	Test if two nodes are connected by a chain of RST nuclei (non-multinuclear). (ll. 65–75)
`traverse_nucleus_chain($nodes)`	Select all subsequently chained RST nuclei (non-multinuclear). (ll. 78–94)
`get_chain($first,$last)`	Select all subsequently chained nodes. (ll. 97–106)
`is_satellite($node,$name?)`	Test if a given node is a RST satellite (in a given relation). (ll. 111–130)
`has_satellite($node,$name?)`	Test if a given node has a RST satellite (in a given relation). (ll. 133–152)
`has_preceding_satellite($node,$name?)`	Test if a given node has a preceding RST satellite (in a given relation). (ll. 155–174)
`get_preceding_satellite($nodes)`	Select all preceding RST satellites. (ll. 177–184)
`has_following_satellite($node,$name?)`	Test if a given node has a following RST satellite (in a given relation). (ll. 187–206)
`get_following_satellite($nodes)`	Select all following RST satellites. (ll. 209–216)
`is_multinuclear($node,$name?)`	Test if a given node is in (a given) multinuclear RST relation to its co-nuclei. (ll. 221–240)
`get_multinuclear_group($node)`	Select all multinuclear related RST nuclei in order of their according group. (ll. 243–256)

Table 4.6.: *XSLT function library for RST validation*

4.3.3. Genre-specific Content Transformation

The second step of the generic to genre-specific document transformation (see Fig. 4.14, p. 77) transforms the validated content structure to its genre-specific output document structures via XSLT. Preferably the XSLT stylesheet uses the same XPath expressions as the Schematron schema from the preceding genre validation (see Sec. 4.3.2), but specifies the content mode choices and layout decisions for these key content structures. To mimic common inline XML processing, the standoff to inline markup transformation pipeline from Section 4.2 is embedded as a pre-transformation within the root-template of a starter XSLT stylesheet. The (sub)genre-specific transformation definitions can be defined by importing and extending the starter stylesheet in its own genre-specific XSLT stylesheet.

To ease the genre-specific transformation definitions, the starter stylesheet provides the following templates and functions. Besides the embedded standoff to inline markup transformation pipeline (see Appx. C.3.3, ll. 20–31), the starter stylesheet defines named templates and functions to process only content ("`content-nodes`", see Appx. C.3.3, ll. 36–41, 52–59) or structure nodes ("`structure-nodes`", see Appx. C.3.3, ll. 44–49, 62–69) within a given context, along with functions to test if the given context itself is a content ("`is-content-node`", see Appx. C.3.3, ll. 72–79) or a structure node ("`is-structure-node`", see Appx. C.3.3, ll. 82–89). Furthermore, the stylesheet provides tunnel parameters (see Kay 2007, Sec. 10.1.2) to control the content output. Specifically, these parameters allow to insert XML nodes as directly preceding or following siblings or as wrapping elements of processed content nodes ("`insert-before`", "`insert-after`", "`wrapper`", see Appx. C.3.3, ll. 94–118). The content mode choices for given content nodes are controlled by a further tunnel parameter as previously described in Section 4.2.3 ("`content-mode`", see Appx. C.1.4, ll. 88–119).

4.3.4. Final Markup Cleanup

Just as in the final standoff to inline markup transformation step described in Section 4.2.3, the genre-specific transformation step is concluded by a XML namespace reorganization (see Appx. C.1.5). Depending on the specific publication channel, further transformation steps might follow to finalize the document processing (e.g. to produce a printable Portable Document Format (PDF) file). The starter stylesheet as a whole is listed in Appendix C.3.3. An example definition of a specific genre and its content transformation follows.

```
<group xmlns="http://www.sfu.ca/rst">
  <segment relname="joint" reltype="multinuc">u1</segment>
  <segment relname="joint" reltype="multinuc">u2</segment>
</group>
```

Listing 4.29.: *RST structure of the example document*

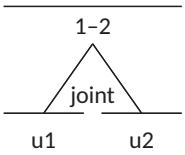


Figure 4.16.: *Schematic representation of the RST structure described in Listing 4.29*

4.3.5. An Example

This section shows an example Schematron-based definition of the "Yojijukugo"-Genre that describes four-character idioms like the "killing two birds with one stone" example from the preceding Section 4.1.6 and according XSLT-based subgenre-specific transformations that create Japanese and English representations of "Yojijukugo"-documents. I.e. the main genre defines the key characteristics of the content structure by means of XPath expressions that are provided as a Schematron schema; based on these key expressions the subgenres define how the content structure is represented for a specific publishing channel.

Content Structure

The content structure defines a specific RST-based organization of the content base's basic units by means of a reference graph that is described via XLink. Comparable to the content base, the RST structure and labels are described within XHTML lists as their container structures, again, and saved in their own files, each. Listing 4.29 above shows the inline XML description of the document's RST structure that distinguishes two content segments that are grouped together in a multinuclear "joint" relation. The first segment represents the basic unit `u1`, the second segment `u2`. Figure 4.16 above shows the graphical representation of the RST instance. Listing 4.30 on the next page shows the standoff XLink-based representation of the RST structure, where the grouping node's resource description is annotated with

```
<ul id="struct">
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
    <span xlink:type="resource" xlink:label="parent" id="node_1"></span>
    <span xlink:type="locator" xlink:label="child" xlink:href="#node_1.1"></span>
    <span xlink:type="locator" xlink:label="child" xlink:href="#node_1.2"></span>
  </li>
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
    <span xlink:type="resource" xlink:label="parent" id="node_1.1"></span>
    <span xlink:type="locator" xlink:label="child" id="content_1.1.1"
          xlink:href="Birds.basic_units.html#u1"></span>
  </li>
  <li xlink:type="extended">
    <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
          xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
    <span xlink:type="resource" xlink:label="parent" id="node_1.2"></span>
    <span xlink:type="locator" xlink:label="child" id="content_1.2.1"
          xlink:href="Birds.basic_units.html#u2"></span>
  </li>
</ul>
```

Listing 4.30.: *RST document structure of the example document (`Birds.RST.struct.html`)*

the unique identifier "`node_1`" and the segments with "`node_1.1`" and "`node_1.2`". The RST structure labeling is described via a XLink linkbase, where Listing 4.31 on the next page shows the XML namespace labeling ("`http://www.sfu.ca/rst`" for root node `node_1`, Listing 4.32 on the next page shows the XML element labeling ("`rst:group`" for `node_1`, "`rst:segment`" for `node_1.1` and `node_1.2`), and Listing 4.33 on page 86 shows the XML attribute labeling ("`rst:relname="joint"`" and "`rst:reltype="multinuc"`" for `node_1.1` and `node_1.2`). The full XHTML files that describe the document's standoff structuring and labeling are listed in Appendix B.1.2.

```
8  <ul xml:base="Birds.RST.struct.html" id="labels">
9    <li xlink:type="extended">
10     <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
             xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
11     <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
             xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
12     <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1"
             xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
13     <span xlink:type="resource" xlink:label="type"
             xlink:role="http://www.w3.org/2001/XMLSchema#QName">xmlns:rst</span>
14     <span xlink:type="resource" xlink:label="value"
             xlink:role="http://www.w3.org/2001/XMLSchema#string">http://www.sfu.ca/rst</span>
15   </li>
     ...
43 </ul>
```

Listing 4.31.: *RST namespace labels of the example document (`Birds.RST.labels.html`)*

```
8  <ul xml:base="Birds.RST.struct.html" id="labels">
   ...
16   <li xlink:type="extended">
17     <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
             xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
18     <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1"
             xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
19     <span xlink:type="resource" xlink:label="type"
             xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:group</span>
20   </li>
21   <li xlink:type="extended">
22     <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
             xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
23     <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1.1"
             xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
24     <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1.2"
             xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
25     <span xlink:type="resource" xlink:label="type"
             xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:segment</span>
26   </li>
     ...
43 </ul>
```

Listing 4.32.: *RST element labels of the example document (`Birds.RST.labels.html`)*

```
8  <ul xml:base="Birds.RST.struct.html" id="labels">
   ...
27   <li xlink:type="extended">
28     <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
             xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
29     <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
             xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
30     <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1.1"
             xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
31     <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1.2"
             xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
32     <span xlink:type="resource" xlink:label="type"
             xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:relname</span>
33     <span xlink:type="resource" xlink:label="value"
             xlink:role="http://www.w3.org/2001/XMLSchema#string">joint</span>
34   </li>
35   <li xlink:type="extended">
36     <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
             xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
37     <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
             xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
38     <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1.1"
             xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
39     <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1.2"
             xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
40     <span xlink:type="resource" xlink:label="type"
             xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:reltype</span>
41     <span xlink:type="resource" xlink:label="value"
             xlink:role="http://www.w3.org/2001/XMLSchema#string">multinuc</span>
42   </li>
43 </ul>
```

Listing 4.33.: *RST attribute labels of the example document (`Birds.RST.labels.html`)*

Multistructured Document Description

The multistructured document description puts all files of the content base and the content structure in context to each other via XLink. The list of basic units is defined as linkbase for the lists of language-specific content realizations and the content labels as linkbase for the

```
8  <div xlink:type="extended">
9    <!-- Structure: RST -->
10   <span xlink:type="arc" xlink:from="struct.RST" xlink:to="labels.RST" xlink:show="embed"
           xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"></span>
11   <span xlink:type="locator" xlink:label="struct.RST"
           xlink:href="Birds.RST.struct.html#node_1"></span>
12   <span xlink:type="locator" xlink:label="labels.RST" xlink:role="http://www.sfu.ca/rst"
           xlink:href="Birds.RST.labels.html#labels"></span>
13   <!-- Content Base -->
14   <span xlink:type="arc" xlink:from="content" xlink:to="units" xlink:show="none"
           xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"></span>
15   <span xlink:type="locator" xlink:label="units"
           xlink:href="Birds.basic_units.html#basic-units"></span>
16   <span xlink:type="locator" xlink:label="content" xlink:href="Birds.content.en.html"
           xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"></span>
17   <span xlink:type="locator" xlink:label="content" xlink:href="Birds.content.ja.html"
           xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"></span>
18 </div>
```

Listing 4.34.: *Multistructured document definition of the example document (`Birds.doc.html`)*

content structure. Listing 4.34 above shows the multistructured document description as a single XLink reference that is embedded in a XHTML `div` element. Appendix B.1.2 lists the full XHTML file and Listing 4.35 on the next page its intermediate inline representation that leads to the final RST structure in the course of the pipeline processing.

Content Validation

The Yojijukugo-Genre defines the key specifics of the document's RST structure by means of a Schematron schema and with the aid of the XSLT function library presented in the preceding Section 4.3.2. In this example the RST structure is restricted to two content segments that must be in the multinuclear "joint" relation to each other. Listing 4.36 on page 89 shows the corresponding Schematron pattern that translates this rule in one positive ("`assert`") and two negative ("`report`") assertions. The first defines the "exactly two children" assertion, the latter ones the "text segments, only" and "multinuclear 'joint' relations, only" assertions. The full Schematron file is listed in Appendix B.1.2.

```
<dag:dag xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags">
  <dag:dag>
    <dag:attribute id="d6e13" type="xmlns:rst"/>
    <dag:attribute id="d6e13" value="http://www.sfu.ca/rst"/>
    <dag:element type="rst:group"/>
    <dag:node>
      <dag:element type="rst:segment"/>
      <dag:attribute id="d6e47" type="rst:relname"/>
      <dag:attribute id="d6e47" value="joint"/>
      <dag:attribute id="d6e64" type="rst:reltype"/>
      <dag:attribute id="d6e64" value="multinuc"/>
      <dag:node>
        <dag:alt>
          <dag:node role="http://www.iana.org/assignments/language-subtag-registry#en">
            one stone</dag:node>
          <dag:node role="http://www.iana.org/assignments/language-subtag-registry#ja">
            &#19968;&#30707;</dag:node>
        </dag:alt>
      </dag:node>
    </dag:node>
    <dag:node>
      <dag:element type="rst:segment"/>
      <dag:attribute id="d6e49" type="rst:relname"/>
      <dag:attribute id="d6e49" value="joint"/>
      <dag:attribute id="d6e66" type="rst:reltype"/>
      <dag:attribute id="d6e66" value="multinuc"/>
      <dag:node>
        <dag:alt>
          <dag:node role="http://www.iana.org/assignments/language-subtag-registry#en">
            two birds</dag:node>
          <dag:node role="http://www.iana.org/assignments/language-subtag-registry#ja">
            &#20108;&#40165;</dag:node>
        </dag:alt>
      </dag:node>
    </dag:node>
  </dag:dag>
</dag:dag>
```

Listing 4.35.: *Intermediate representation of the example document*

Content Transformation (I)

A first Yojijukugo-Subgenre defines the transformation of according content structures to its Japanese representations via a XSLT stylesheet. In this example the RST document

```
<pattern>
  <title>Yojijukugo Genre</title>
  <rule context="/rst:group">
    <assert test="count(*) = 2">
      A "Yojijukugo"'s top-level group has exactly two children.</assert>
    <report test="*[not( self::rst:segment )]">
      A "Yojijukugo"'s top-level group is composed of content segments, only.</report>
    <report test="*[not( rst:is_multinuclear( ., 'joint' ) )]">
      A "Yojijukugo"'s top-level group is composed of nuclei that are in "joint" relation
      to each other, only.</report>
  </rule>
</pattern>
```

Listing 4.36.: *Schematron schema defining the Yojijukugo-Genre (`Yoji.content.sch`)*

```
<xsl:template match="/rst:group">
  <xsl:element name="html:p">
    <xsl:apply-templates>
      <xsl:with-param tunnel="yes" name="content-mode"
            select="'http://www.iana.org/assignments/language-subtag-registry#ja'"/>
    </xsl:apply-templates>
  </xsl:element>
</xsl:template>
```

Listing 4.37.: *XSLT stylesheet defining the Japanese subgenre (`Yoji.content2form.ja.xsl`)*

is transformed to XHTML. Listing 4.37 above shows the XSLT template that defines the transformation for the top-level `rst:group` element that was defined as the genre's key structure. All its descendants are wrapped in one common `html:p` element, all its descendant content nodes output in Japanese content mode. The transformation of the "killing two birds with one stone" document described previously in this section would produce a paragraph with the text "一石二鳥". Appendix B.1.2 lists the full XSLT stylesheet that describes the Japanese-Subgenre transformation and its XProc definition.

Content Transformation (II)

A second Yojijukugo-Subgenre defines similar to the Japanese subgenre the transformation to an English representation. Listing 4.38 on the next page shows the according XSLT

```
39 <xsl:template match="/rst:group">
40   <xsl:element name="html:p">
41     <xsl:apply-templates select="*[2], yoji:separator(), *[1]">
42       <xsl:with-param tunnel="yes" name="content-mode"
             select="'http://www.iana.org/assignments/language-subtag-registry#en'"/>
43     </xsl:apply-templates>
44   </xsl:element>
45 </xsl:template>
```

Listing 4.38.: *XSLT stylesheet defining the English subgenre (`Yoji.content2form.en.xsl`)*

stylesheet that is almost identical to the Japanese subgenre, but interchanges the first and second text segment and inserts a comma between them. The content transformation is set to English content mode, leading to the output text "two birds, one stone" for the given example document. Appendix B.1.2 lists the full XSLT stylesheet that describes the English-Subgenre transformation and its XProc definition.

4.3.6. Discussion

The previous sections have introduced a description and processing framework that translates the formal genre-aware document model from the preceding Chapter 3 to a technical implementation. The framework describes a genre by means of a Schematron schema that specifies the key compositions of a document's primary structure and by means of according XSLT stylesheets that describe the relations between the primary structure towards its co-structures. Both languages, Schematron and XSLT, use XPath to identify specific document structures. Therefore, the key structure specifications within the genre validation are easily transferred and reused within the subgenre transformation.

The UML diagram in Figure 4.17 on the next page gives an overview of the framework architecture that transforms a genre-specific primary RST-based structure to its corresponding co-structures. The genre-specific descriptions are specified by means of a Schematron schema for the superior genre and by means of XSLT stylesheets for the subgenre transformations within a shared XProc pipeline. Appendix C.3.1 shows a generalized example instance of a genre-specific XProc pipeline. As the genre validation step consists of compound sub-steps again, it is moved to its own XProc definition file that is listed in Appendix C.3.2. The genre validation step can be specified by means of pre-defined RST-based schema assertions and

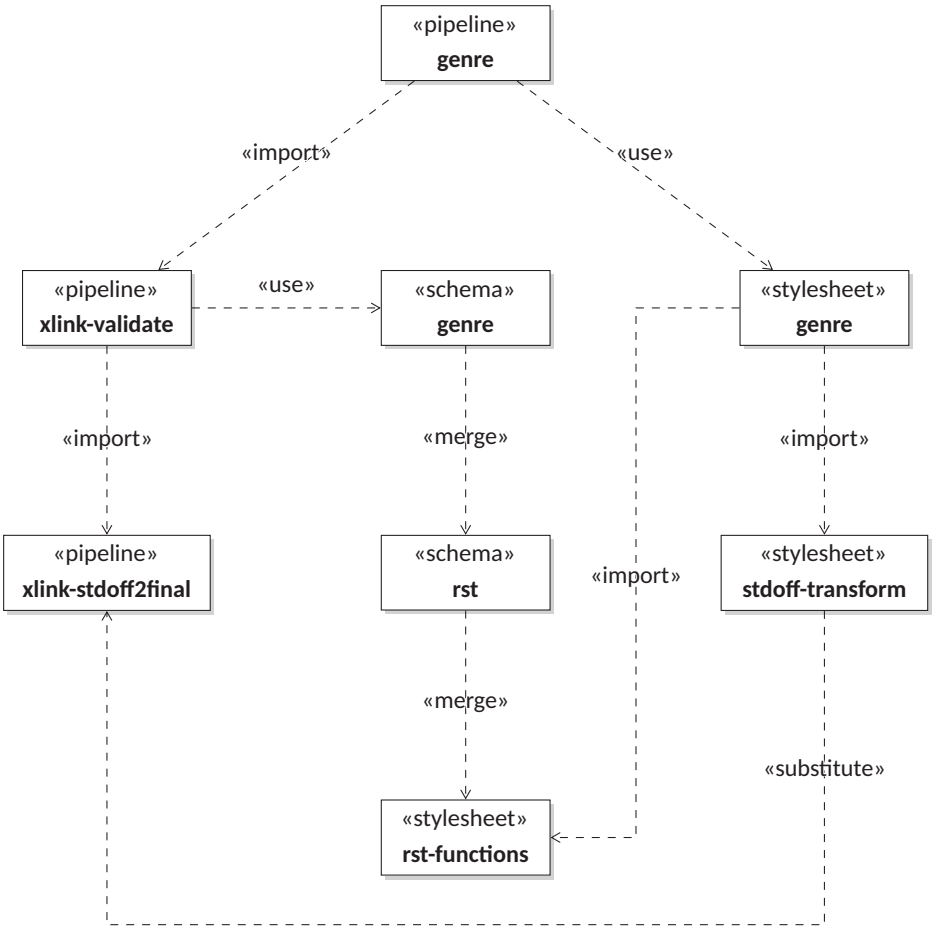


Figure 4.17.: *Schematic diagram of the genre processing components*

XPath functions as explained in the preceding Section 4.3.2. The genre-specific transformations can be defined based on a XSLT stylesheet that integrates the standoff to inline markup transformation to common XSL processing as explained in the preceding Section 4.3.3.

In the context of productive publishing, the document's primary structure is described via RST, as it represents the author's view on the document in the process of creating consistent information. As the RST-based description of a document is independent of its output realization, different sets of XSLT stylesheets can be provided to define alternative subgenre-specific document realizations and therefore realize a multichannel publication of a given document.

4.4. Conclusion

With the preceding sections a technical implementation of the genre-aware document model was introduced that relies exclusively on established publishing standards. The DAG defined by the formal multistructured document model is described by means of XLink where the graph semantics are specified via RDF Schema for the arcs and via XML Schema for the resources. The reconstruction of the specific document structures as common inline XML is realized by means of consecutive XSLT steps that are orchestrated via XProc. The definition of genre-specific restrictions is described via Schematron for the composition of the source structure and via XSLT for the structure transformations, where both—Schematron and XSLT—rely on XPath to specify the characteristic structures.

With these common publishing standards the framework relies on widespread processing tools only. Missing processor features (e.g. automated XLink traversal) were added by means of software independent description languages (e.g. XSLT). Thus, the framework remains independent of a specific proprietary software product. Furthermore, it was paid attention to provide a framework implementation that is executable on free available software (e.g. relating to XProc and XSLT processors).

The framework realizes the description of document-specific labeled structures and content modes (i.e. the document types of distinct document structures within specific namespaces) by means of a declarative approach. Thus, the document processing within the framework was realized on top of a generic document description logic that is document type independent as well. As a consequence the framework is not restricted to any document type.

The following chapter demonstrates the framework's usability in the context of a multichannel scenario by means of an example application.

La inteligencia sin ambición es como un pájaro sin sus alas.

— Salvador Dalí

5. A Multichannel Example

This chapter demonstrates the usability of the previously introduced document description and processing framework by example of a bird fieldguide and different output channels. Section 5.1 describes the example document by means of a multistructured XML document, where the content base includes textual and graphical realizations that a document structure organizes via RST. Section 5.2 describes the document's characteristics by means of a genre definition that is further differentiated into subgenres that specify the different output channels via alternative structure transformations. The chapter describes three alternative output channels that transform the RST structure of the document into corresponding output structures representing print, movie, and 3D derivatives. Finally, Section 5.3 concludes the example application.

5.1. Document Description

This section describes an example page from a bird fieldguide as a XLink-based multistructured document. The example page provides information about the gannet—a seabird. This example has been introduced in Allen et al. (1999) with special regard to the genre-specific composition of the page and more generally in Henschel et al. (2002) with regard to an automated layout generation. Figure 5.1 on the next page shows the example page and its content segmentation into basic units based on the following distinguishable visual groups. Units 1–3 represent the document's title and subtitle, or more specific, the bird's name and family. Units 4 and 5 provide information about the bird's presence in Britain and size by means of small diagrams. Units 6–13 represent a basic description of the bird's lifestyle by means of full sentences. Units 14–20 represent a fact file that provides further information about the bird by means of a table. Unit A provides a photographic image showing the bird: Units B and C provide information about breeding places and different age stages by means

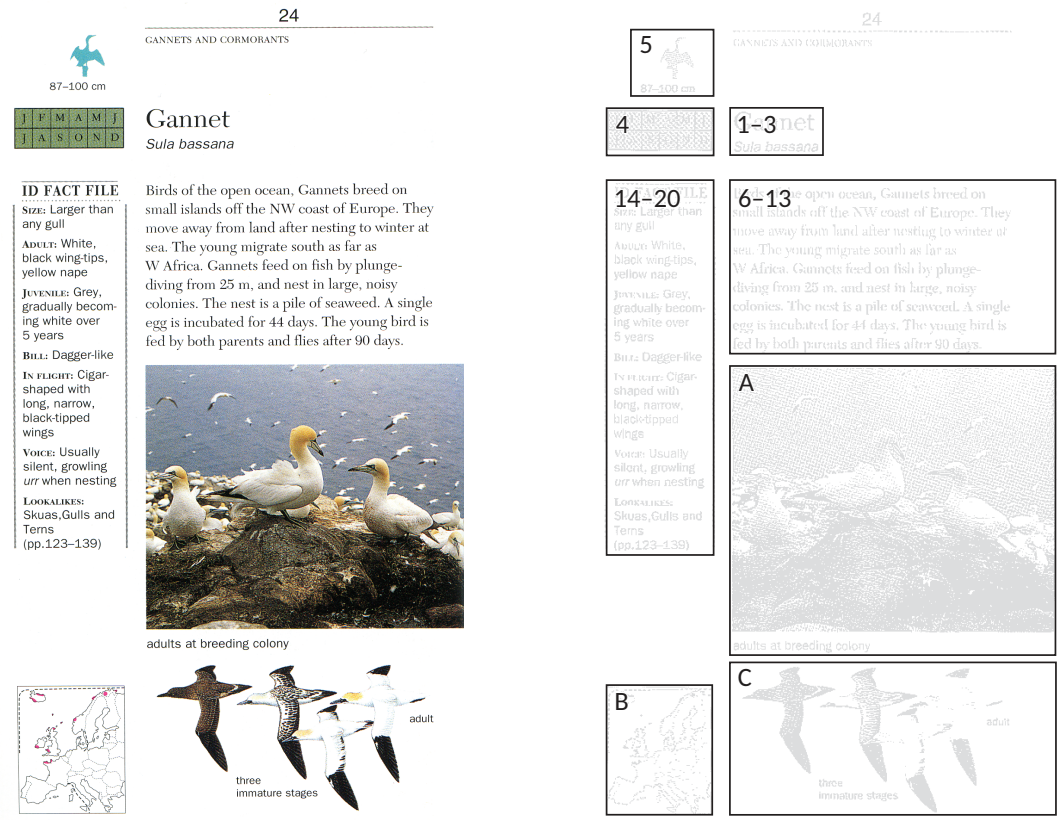


Figure 5.1.: *Page of an example bird fieldguide (a) as originally printed in Holden (1996, p. 24) and (b) its content distribution (see Table 5.1)*

of diagrams. Correspondingly, Table 5.1 on the next page shows the twenty basic units that describe the specific content fragments of the example page by means of alternative textual, elliptic, and tabular realizations.

The specific description of the example as a multistructured document is given in the following sections. Section 5.1.1 shows the XLink-based description of the content base, Section 5.1.2 the description of the RST-based content structure, and Section 5.1.3 the multistructured document description that unites all the document's components.

no.	**text**—"full"	**list**—"ellipsis"	**table**—"col. 1" | "col. 2"
1	[The] Gannet	[Gannet]	[*Name* | Gannet]
2	Gannet is in Latin Sula Bassana.	Sula bassana	[*Latin*] | Sula bassana
3	The Gannet belongs to the Sulidae.	Family Sulidae	[*Family*] | Sulidae
4	The gannet is in Britain from January till December.	In Britain January–December	*In Britain* | Jan–Dec
5	The gannet is 87–100cm long.	87–100cm	*Size* | 87–100cm
6	Birds of the open ocean, Gannets breed on small islands off the [North-West] coast of Europe.	Breed on small islands off the NW coast of Europe	*Breeding* | On small islands off the NW coast of Europe
7	They move away from land after nesting to winter at sea.	Move away from land after nesting to winter at sea	*Habitat* | Move away from land after nesting to winter at sea
8	The young migrate south as far as [West] Africa.	Young migrate south as far as W Africa	*Migration* | Young migrate south as far as W Africa
9	Gannets feed on fish by plunge-diving from 25m.	Feed on fish by plunge-diving from 25m	*Food* | Fish. Plunge-diving from 25m
10	They nest in large, noisy colonies.	Nest in large noisy colonies	*Habitat* | Large noisy colonies
11	The nest is a pile of seaweed.	Nest pile of seaweed	*Nest* | Pile of seaweed
12	A single egg is incubated for 44 days.	Single egg incubated for 44 days	*Egg* | A single egg, incubated for 44 days
13	The young bird is fed by both parents and flies after 90 days.	Young fed by both parents, flies after 90 days	*Parenting* | Young fed by both parents, flies after 90 days
14	The gannet is larger than any gull.	Larger than any gull	*Size* | Larger than any gull
15	The adult is white, has black wing-tips, and a yellow nape.	Adult white, black wing-tips, yellow nape	*Adult* | White, black wing-tips, yellow nape
16	The juvenile gannet is grey, gradually becoming white over 5 years.	Juvenile grey, gradually becoming white over 5 years	*Juvenile* | Grey, gradually becoming white over 5 years
17	The bill is dagger-like.	Bill dagger-like	*Bill* | Dagger-like
18	In flight, the gannet is cigar-shaped with long, narrow, black-tipped wings.	In Flight, cigar-shaped with long, narrow, black-tipped wings	*In flight* | Cigar-shaped with long, narrow, black-tipped wings
19	The gannet is usually silent, growling ["urr"] when nesting.	Usually silent, growling ["urr"] when nesting	*Voice* | Usually silent, growling ["urr"] when nesting
20	Lookalikes are Skuas, Gulls and Terns [. . .].	Lookalikes Skuas, Gulls and Terns [. . .]	*Lookalikes* | Skuas, Gulls and Terns

Table 5.1.: *Content realizations of the fieldguide example shown in Figure 5.1 as defined in the original source material (cf. Appx. A.6.1) with slight adaptions ("[. . .]")*

```
<ul>
  <li id="t1">The Gannet </li>
  <li id="t2">Gannet is in Latin Sula Bassana. </li>
  <li id="t3">The Gannet belongs to the Sulidae. </li>
  ...
</ul>
```

Listing 5.1.: *Text realizations of the fieldguide content encoded in XHTML*

```
<ul>
  <li id="l1">Gannet</li>
  <li id="l2">Sula bassana</li>
  <li id="l3">Family Sulidae</li>
  ...
</ul>
```

Listing 5.2.: *List realizations of the fieldguide content encoded in XHTML*

```
<table>
  <tr id="t1"> ↲ <th>Name</th> ↲ <td>Gannet</td> ↲ </tr>
  <tr id="t2"> ↲ <th>Latin</th> ↲ <td>Sula bassana</td> ↲ </tr>
  <tr id="t3"> ↲ <th>Family</th> ↲ <td>Sulidae</td> ↲ </tr>
  ...
</table>
```

Listing 5.3.: *Table realizations of the fieldguide content encoded in XHTML*

```
<ul>
  <li id="i1"><img alt="Adults at breeding colony" src="media/images/gannet.jpg"/></li>
  <li id="i6"><img alt="map" src="media/images/holden-map.jpg"/></li>
  <li id="i16"><img alt="three-age-stages" src="media/images/holden-drawing.jpg"/></li>
</ul>
```

Listing 5.4.: *Image realizations of the fieldguide content encoded in XHTML*

5.1.1. Content Base

The content base provides alternative realizations for the twenty basic units of the gannet example. The basic units are described within a XLink linkbase by means of the interlinked realizations. The list of basic units is shown in Appendix B.2.1. Listings 5.1–5.4 above

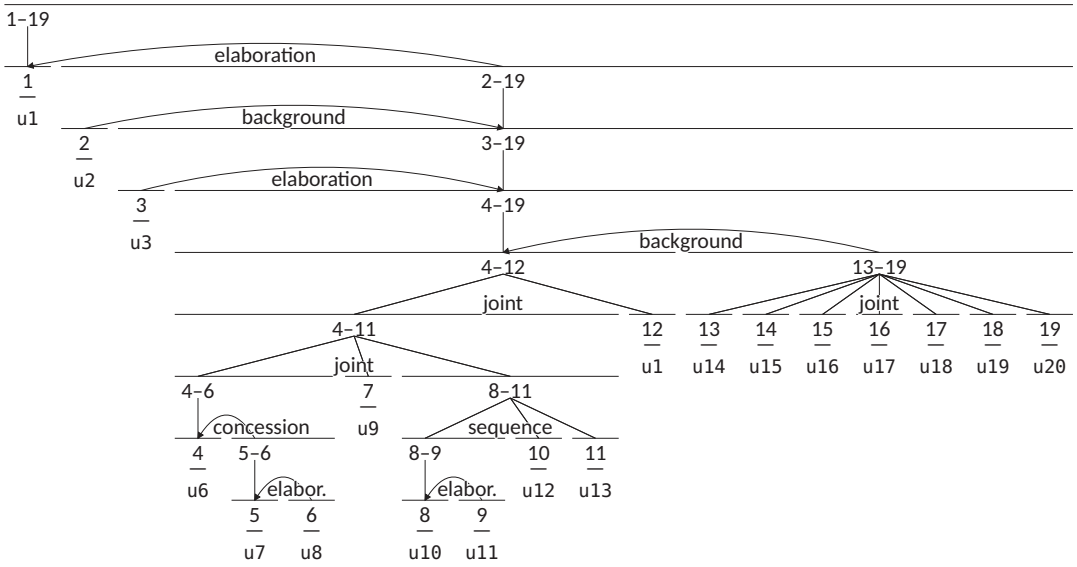


Figure 5.2.: *RST structure of the Gannet example*

show the first entries of the content realization lists that are grouped and separated by content mode. Here, the content realizations provided by the source material from Henschel et al. (2002) that originally distinguishes the content modes “full”, “ellipsis”, “table-form”, and “graphics” are reorganized by means of the XHTML “text”, “list”, “tables”, and “image” modules (Austin et al. 2010). Note, that in the content base the units A, B, and C (bird, breeding place, and age stages) from the original page segmentation in Figure 5.1 on page 94 are described as alternative realizations of units 1, 6, and 16. The full listing of the content realization descriptions is given in Appendix B.2.1, the original source material in Appendix A.6.1.

5.1.2. Content Structure

The content is organized by means of a RST tree and described by means of XLink-based lists of referenced structure nodes and labels. Listing 5.5 on the next page shows the inline representation of the specified XML instance and Figure 5.2 above the described RST tree with the following embedded basic units. The RST content segment 1 represents the title realized by unit 1. Segments 2 and 3 represent the subtitle realized by units 2 and 3.

```
<group xmlns="http://www.sfu.ca/rst">
  <segment>u1
    <group relname="elaboration" reltype="rst">
      <group>
        <segment relname="background" reltype="rst">u2</segment>
        <group>
          <segment relname="elaboration" reltype="rst">u3</segment>
          <group>
            <group relname="joint" reltype="multinuc">
              <group relname="joint" reltype="multinuc">
                <segment>u6
                  <group relname="concession" reltype="rst">
                    <segment>u7
                      <segment relname="elaboration" reltype="rst">u8</segment>
                    </segment>
                  </group>
                </segment>
              </group>
              <segment relname="joint" reltype="multinuc">u9</segment>
              <group relname="joint" reltype="multinuc">
                <group relname="sequence" reltype="multinuc">
                  <segment>u10
                    <segment relname="elaboration" reltype="rst">u11</segment>
                  </segment>
                </group>
                <segment relname="sequence" reltype="multinuc">u12</segment>
                <segment relname="sequence" reltype="multinuc">u13</segment>
              </group>
            </group>
            <segment relname="joint" reltype="multinuc">u1</segment>
            <group relname="background" reltype="rst">
              <segment relname="joint" reltype="multinuc">u14</segment>
              <segment relname="joint" reltype="multinuc">u15</segment>
              <segment relname="joint" reltype="multinuc">u16</segment>
              <segment relname="joint" reltype="multinuc">u17</segment>
              <segment relname="joint" reltype="multinuc">u18</segment>
              <segment relname="joint" reltype="multinuc">u19</segment>
              <segment relname="joint" reltype="multinuc">u20</segment>
            </group>
          </group>
        </group>
      </group>
    </group>
  </segment>
</group>
```

Listing 5.5.: *RST structure of the example document*

```
8  <div xlink:type="extended">
9    <!-- Structure: RST -->
10   <span xlink:type="arc" xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
           xlink:show="embed" xlink:from="struct.RST" xlink:to="labels.RST"></span>
11   <span xlink:type="locator" xlink:label="struct.RST"
           xlink:href="Gannet.RST.struct.xhtml#node_1"></span>
12   <span xlink:type="locator" xlink:label="labels.RST" xlink:role="http://www.sfu.ca/rst"
           xlink:href="Gannet.RST.labels.xhtml#labels"></span>
13   <!-- Content Base -->
14   <span xlink:type="arc" xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
           xlink:show="none" xlink:from="content" xlink:to="units"></span>
15   <span xlink:type="locator" xlink:label="units"
           xlink:href="Gannet.basic_units.xhtml#basic-units"></span>
16   <span xlink:type="locator" xlink:label="content" xlink:href="Gannet.content.text.xhtml"
           xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
17   <span xlink:type="locator" xlink:label="content" xlink:href="Gannet.content.list.xhtml"
           xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
18   <span xlink:type="locator" xlink:label="content" xlink:href="Gannet.content.table.xhtml"
           xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
19   <span xlink:type="locator" xlink:label="content" xlink:href="Gannet.content.img.xhtml"
           xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod"></span>
           ...
23 </div>
```

Listing 5.6.: *Multistructured document definition of the example document (`Gannet.doc.xhtml`)*

Segments 4–11 represent the basic description realized by units 6–13. Segment 12 represents the photograph originally referring to unit A (see Fig. 5.1, p. 94) that is described within the content base as an alternative realization of unit 1. Finally, segments 13–19 represent the fact sheet realized by units 14–20. Note, that according to the original RST description (cf. Appx. A.6.2) units 4 and 5 are not used in the document structure within the example at hand. The XLink-based description of the RST tree by means of a distinct structuring and distinct labeling is listed in Appendix B.2.1, the original RST description within the source material from Henschel et al. (2002) in Appendix A.6.2.

5.1.3. Multistructured Document

All components of the example document, i.e. the content base and the document structure, are related to each other within the multistructured document description by means of one single link. Listing 5.6 above shows the multistructured document with the RST

```
12 <pattern>
13   <title>Fieldguide Genre</title>
14   <rule context="/rst:group">
15     <assert test="(count(*) = 1) and rst:segment[rst:is_nucleus(.)]">
16       A "Fieldguide"'s top-level group is composed of exactly one content segment nucleus.
         </assert>
17   </rule>
18   <rule context="/rst:group/rst:segment">
19     <assert test="(count(*) = 1) and rst:group[rst:is_satellite(.,'elaboration')]">
20       A "Fieldguide"'s initial content segment has exactly one exclusive "rst:group"
         satellite in "elaboration" relation. </assert>
21   </rule>
22   <rule context="/rst:group/rst:segment/rst:group">
23     <assert test="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]">
24       Traversing the "Fieldguide"'s initial satellite's path of descending nuclei leads to
         a "rst:group" consisting of two multinuclear nodes that are in "joint" relation and
         that has a satellite that is in "background" relation to its nucleus. </assert>
25     <report test="rst:get_chain( ., rst:traverse_nucleus_chain(.)[
         rst:core-structure(.)]/parent::rst:* )[rst:has_following_satellite(.)]">
26       The traversed nuclei starting from the "Fieldguide"'s first layer satellite don't
         have following satellites. </report>
27   </rule>
28 </pattern>
```

Listing 5.7.: *Schematron schema defining the Fieldguide-Genre (`Fieldguide.content.sch`)*

structuring embedded in lines 9–12 and the content base in line 13 ff. The full listing of the multistructured document description is provided in Appendix B.2.1.

5.2. Genre Description

This section defines the common key structures of the bird fieldguide example page from the preceding section as a set of XPath-based Schematron assertions (see Sec. 4.3). While these assertions constitute a superordinate “fieldguide” genre, the specific XPath expressions are used in Sections 5.2.1–5.2.3 to define subgenre-specific XSLT stylesheets to transform the content structure into alternative output structures.

First, the Schematron schema for the superordinate fieldguide genre is described, that specifies four restrictions for the content structure. Listing 5.7 above shows the Schematron

pattern that specifies the restrictions for the fieldguide genre. These restrictions are specified by means of XPath expressions that use the RST-specific function library introduced in Section 4.3.2 before. The full Schematron schema is listed in Appendix B.2.2. The four restrictions are explained in the following and will be referenced under the names R1–R4 in subsequent sections:

- R1: The first assertion (Lst. 5.7, ll. 14–17) restricts the content to start with a single nucleus (seg. 1 / `u1` in Fig. 5.2, p. 97). This nucleus represents the original page title of the example page (Fig. 5.1a, p. 94).
- R2: As the further content provides detail on the intial nuleus' subject (e.g. a specific bird), the second assertion (Lst. 5.7, ll. 18–21) declares that the entire further content is attached to the initial nucleus as a single elaboration satellite (seg. 2–19 in Fig. 5.2, p. 97).

The following two assertions (Lst. 5.7, ll. 22–27) identify a "core structure" within the detailed bird description that is specified as follows:

- R3: The core structure is composed of three items, where the first two items are related to each other by means of a multinuclear joint relation (seg. 4–12 in Fig. 5.2, p. 97) and the third item constitutes a background satellite (seg. 13–19 in Fig. 5.2, p. 97). The two nuclei represent the lifestyle specifics and the photograph that are the core information of a fieldguide entry. The satellite represents the fact-file that provides supplementary information about the bird.
- R4: The last assertion (Lst. 5.7, ll. 25 f.) declares that the core structure has only preceding satellites (e.g. seg. 2 / `u2` and seg. 3 / `u3` in Fig. 5.2, p. 97) following the starting nucleus (see R1). These satellites represent additional idiosyncratic information to the initial nucleus' subject as e.g. realized in the original page by means of a subtitle (Fig. 5.1a, p. 94).

These four key restrictions are used in the following sections to specify three subordinated subgenres. Section 5.2.1 specifies a publication channel that realizes the fuildguide as printed matter, Section 5.2.2 a channel for a movie realization, and Section 5.2.3 for 3D respectively.

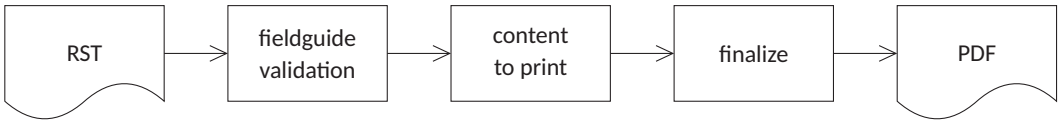


Figure 5.3.: *Fieldguide print pipeline*

5.2.1. Print-specific Channel

The first fieldguide subgenre example specifies a derivative representation of the content on a page within a printed field manual. This output channel follows the layout decisions made in the original example from Henschel et al. (2002) and is realized by means of a XProc pipeline. Figure 5.3 above shows the pipeline with its substeps that transforms the RST-based content structure into a printable PDF file. The actual XProc implementation of the pipeline is listed in Appendix B.2.2 on pages 184 f. and describes the following steps. First, the given multistructured document is transformed into its inline representation and validated against the general fieldguide restrictions introduced in the preceding section. Next, the RST description of the document is transformed into an output description by means of a XSLT stylesheet. More specifically, the document's RST representation is transformed into a XHTML representation, where for each key structure restriction R1–R4 a template decides on the output structure and the descending nodes' content mode. The stylesheet implementation is listed in Appendix B.2.2 on pages 185 ff. and specifies the following transformations that are based on the considerations given by Henschel et al. (2002):

- The nucleus specified by R1 is output as the page heading as it represents the major subject of the fieldguide entry. Therefore, the nucleus is transformed into a `html:h1` element that is realized by means of the "list" mode.
- The preceding satellites of the core structure specified by R4 are particularly output as they represent the classification information about the document's subject. Specifically, these contents are emphasized via `html:em` and separated via `html:br` elements. The specific subject classifications are realized by means of the "list" mode.

The core structure specified by R3 distinguishes three items that are transformed as follows:

- The first item represents the basic description of the fieldguide's subject and is output as full text. Therefore, the first item is wrapped into a `html:p` element and realized by means of the "text" mode.

Gannet

Sula bassana
Family Sulidae

Birds of the open ocean, Gannets breed on small islands off the NW coast of Europe. They move away from land after nesting to winter at sea. The young migrate south as far as W Africa. Gannets feed on fish by plunge-diving from 25m. They nest in large, noisy colonies. The nest is a pile of seaweed. A single egg is incubated for 44 days. The young bird is fed by both parents and flies after 90 days.

Size	Larger than any gull
Adult	White, black wing-tips, yellow nape
Juvenile	Grey, gradually becoming white over 5 years
Bill	Dagger-like
In flight	Cigar-shaped with long, narrow, black-tipped wings
Voice	Usually silent, growling urr when nesting
Lookalikes	Skuas, Gulls and Terns

(a)

Herring Gull

Larus argentatus
Family Laridae

A familiar gull around the coasts of NW Europe. It is seldom seen far out to sea and also visits inland rubbish tips, farmland and parks. It roots on the sea along sheltered coasts and on reservoirs. Herring Gulls eat a variety of food including carrion and offal from fishing boats. They nest on open, often sloping ground and sometimes on buildings. The 3 eggs hatch after 28 days. Young leave the nest after 3 days and fly at about 35 days.

Size	Larger than Black-headed Gull
Adult	Pearl-grey back, fierce eye with yellow iris
Juvenile	Mottled brown, bill becoming paler and back greyer
Bill	Powerful, with hooked tip. Yellow with red spot
In flight	Broad wings, heavy-looking
Voice	Wailing, laughing, crying
Lookalikes	Common Gull, Lesser Black-backed Gull

(b)

Figure 5.4.: *Print-specific output of (a) the fieldguide document described in Listing 5.6*[1] *and of (b) an alternative document*[2] *of the same genre (Holden 1996, p. 132)*

- The second item represents the portrayal of the fieldguide's subject. Therefore, the second item is wrapped into a `html:figure` element and realized by means of the "image" mode.
- The third item represents the supplementary fact-file and is output in table form. Thus, the third item is transformed into a `html:table` element with its content nodes wrapped into `html:tr` elements and realized by means of the "table" mode.

The resulting XHTML output from the content transformation of the example fieldguide is listed in Appendix B.2.2 on pages 187 f. and further processed. In order to create a printable

1. Image source: Andreas Trepte. *Morus bassanus — Northern Gannet.* Wikimedia Commons, May 2009. `https://commons.wikimedia.org/wiki/File:Morus_bassanus_adu.jpg` (12.10.2013).
2. Image source: Andreas Trepte. *Larus argentatus — adult Herring Gull.* Wikipedia, October 2005. `https://en.wikipedia.org/wiki/File:Larus_argentatus_ad.jpg` (05.03.2015).

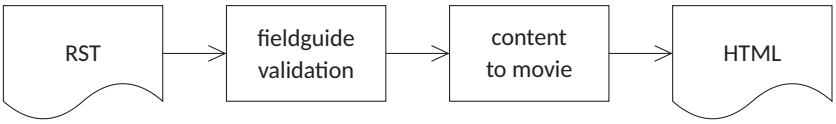


Figure 5.5.: *Fieldguide movie pipeline*

document, the logical output structure described by XHTML has to be converted into a physical description—preferably within a PDF file. An example pipeline that uses the TEX typesetting engine (Knuth 1986) to layout XHTML documents and that creates according PDF files has been introduced by Schlupkothen (2014). A corresponding output of the example document as a field manual page is shown in Figure 5.4 on the preceding page.

5.2.2. Movie-specific Channel

The second fieldguide subgenre example specifies a derivative representation of the content as a short video clip that is meant for playback on mobile devices such as smartphones. Figure 5.5 above shows the pipeline with its substeps that transforms the RST-based content structure into an according HTML output. The actual XProc implementation of the pipeline is listed in Appendix B.2.2 on pages 188 f. The specific processing steps are described in the following. After being validated against the general fieldguide schema, the RST description of the document is transformed via XSLT into a subgenre-specific output description. As in the preceding channel example, the output structure and the descending nodes' content mode are defined for each key structure R1–R4 by means of according XSLT templates. The stylesheet implementation that creates the movie-specific HTML output is listed in Appendix B.2.2 on pages 189 ff. and specifies the following transformations:

- The nucleus specified by R1 that represents the major subject of the fieldguide entry is output as a caption and overlays the movie on the top left corner during the entire playback (see Fig. 5.7, p. 107). Specifically, the document as a whole is transformed into a `html:figure` element where the movie caption is represented by means of a `html:figcaption` element and realized via "list" mode.
- The satellites specified by R4 represent additional information about the document's subject and are omitted for the movie output.

The core structure specified by R3 distinguishes the following three items:

```
8  <ul>
9    <li id="a1"><audio><source src="media/audio/1.wav" type="audio/wav"/></audio></li>
10   <li id="a2"><audio><source src="media/audio/2.wav" type="audio/wav"/></audio></li>
11   <li id="a3"><audio><source src="media/audio/3.wav" type="audio/wav"/></audio></li>
     ...
29 </ul>
```

Listing 5.8.: *Audio realizations of the fieldguide content encoded in XHTML*

```
8  <ul>
9    <li id="v6"><video><source src="media/video/6.mp4" type="video/mp4"/></video></li>
10   <li id="v7"><video><source src="media/video/7.mp4" type="video/mp4"/></video></li>
11   <li id="v8"><video><source src="media/video/8.mp4" type="video/mp4"/></video></li>
     ...
17 </ul>
```

Listing 5.9.: *Video realizations of the fieldguide content encoded in XHTML*

- The first item represents the basic description of the fieldguide's subject and is output as a video sequence with a synchronized audio track. Therefore, the first item's descendants are transformed twice: a first sequence is realized by means of the "video" mode and another sequence realized by means of the "audio" mode. HTML5 provides with `html:video` and `html:audio` elements that can be synchronized via `mediagroup` attributes. The successive playback of the document's videos is realized via JavaScript.
- The second item represents the portrayal of the fieldguide's subject and is used as preview image while the movie is stopped. Therefore, the second item is added to the `html:video` elements via `poster` attribute and realized by means of the "image" mode.
- The third item represents a supplementary fact-file that is omitted for the movie output.

As this publishing channel uses audiovisual presentation, the content base has to provide new content realizations that describe the basic units by means of the "audio" and "video" mode. Listings 5.8 and 5.9 above show excerpts of the new content realization descriptions that are added to the document's content base. Appendix B.2.1 on pages 171 f. and 172 show the full audio and video descriptions for the example document as encoded via HTML5. Note, that HTML5 documents do not provide functional document type declarations (see Berjon et al. 2014, Sec. 8.1.1). I.e., HTML5 documents are not related to a specific DTD that could

(a) *6.mp4*[3] (b) *7.mp4*[4] (c) *8.mp4*[5]

(d) *9.mp4*[6] (e) *10.mp4*[7] (f) *11.mp4*[8]

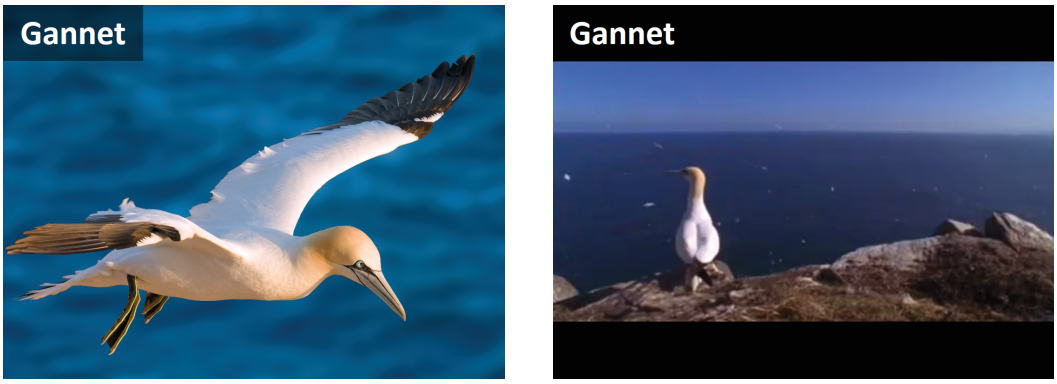
(g) *12.mp4*[9] (h) *13.mp4*[10]

Figure 5.6.: *Thumbnails of the video realizations encoded in Listing 5.9*

declare e.g. attribute types. Hence, a XML processor cannot resolve fragment identifiers, as these references are related to the `ID` attribute type. To solve this problem within the content base's audio and video descriptions the abbreviated HTML5 document type declarations are extended by the following local attribute-list declaration that specifies the attribute type of `li` elements' `id` attributes: `<!DOCTYPE html [<!ATTLIST li id ID #IMPLIED>]>`.

In this example the specific audio realizations were created automatically from the full text realizations (see Tab. 5.1, p. 95) using a Text to Speech (TTS) system. Figure 5.6 above shows the according video realizations that were gathered from a public web video platform. The videos are named according to the related basic units and comprise the following material:

3. Source: SeabirdCentre. *Gannets in flight — A short film about the gannets of the Bass Rock in North Berwick.* Youtube, December 2007. `http://www.youtube.com/watch?v=F30TFQGNFXc` (18.12.2014). Min. 0:00–0:16
4. *Ibid.*, min. 0:27–0:40 5. *Ibid.*, min. 0:54–1:06 6. *Ibid.*, min. 1:08–1:22 7. *Ibid.*, min. 1:22–1:32
8. *Ibid.*, min. 2:21–2:35
9. Source: Toby Shanley. *Gannet chick on Motuora Island.* Youtube, December 2012. `https://www.youtube.com/watch?v=jTpMFz8SC_U` (18.12.2014).
10. Source: moj. *Fou de Bassan, (gannet) Parc national de l'Île-Bonaventure-et-du-Rocher-Percé, Quebec.* Youtube, June 2010. `https://www.youtube.com/watch?v=gdfvygv2QtE` (18.12.2014).



(a) (b)

Figure 5.7.: *Movie-specific output of the fieldguide document described in Listing 5.6 and as rendered by a web browser (a) in preview state and (b) during playback*

- Video `6.mp4` (see Fig. 5.6a) is composed of three successive shots: a long shot on Bass Rock island / a semi long shot on the cliffs / a panning shot on a breading colony.
- Video `7.mp4` (see Fig. 5.6b) is composed of a single full shot on a gannet that glides over the breading colony.
- Video `8.mp4` (see Fig. 5.6c) is composed of three successive shots: a full shot on a gannet departing from the cliff / a full shot on the gannet gliding in front of the cliff / a medium shot on the gannet gliding over the sea.
- Video `9.mp4` (see Fig. 5.6d) is composed of four shots: a full shot on a gannet plunge-diving into the sea (3x) / a medium shot on a gannet catching a fish under the sea.
- Video `10.mp4` (see Fig. 5.6e) is composed of two shots: a shot on a breading colony with flying gannets in the background / a closer panning shot on the breading colony.
- Video `11.mp4` (see Fig. 5.6f) is composed of three shots: a close-up on the feeds of a gannet walking over rocks and seaweed / a shot on the head of a gannet with seaweed in the beak / a medium shot on a gannet that arranges seaweed for the nest.
- Video `12.mp4` (see Fig. 5.6g) is composed of a single shot on a gannet sitting on a nest and regurgitating a fish as a fledgling appears from beneath.
- Video `13.mp4` (see Fig. 5.6h) is composed of a single shot on a feeding gannet.

The resulting XHTML output from the content transformation of the example fieldguide is listed in Appendix B.2.2 on pages 193 ff. The corresponding output as rendered by a web browser is shown in Figure 5.7 above.

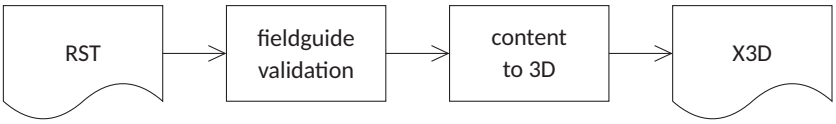


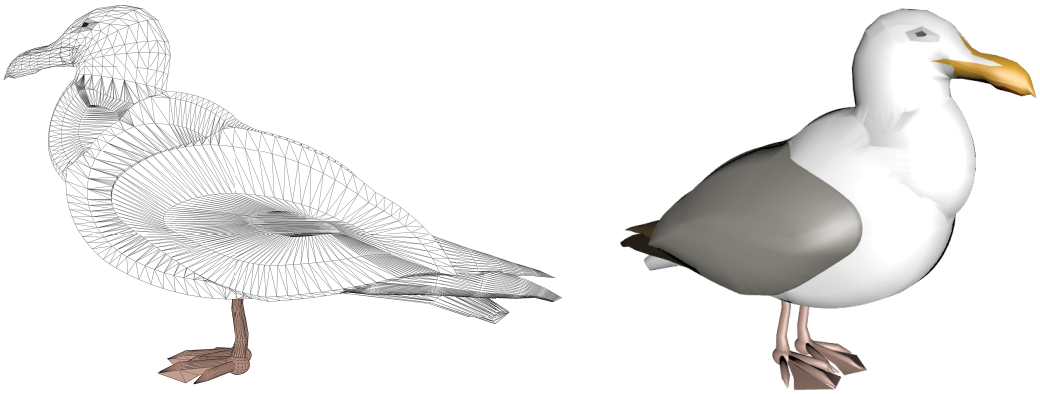
Figure 5.8.: *Fieldguide 3D pipeline*

5.2.3. 3D-specific Channel

The third subgenre example specifies a derivative representation of the fieldguide content as a three dimensional scene that could e.g. be part of a virtual museum of natural science. Specifically, the output scene is composed of a cylindric stand that displays the document's subject and two poster panels that present further information.

Figure 5.8 above shows the pipeline with its substeps that transforms the RST-based content structure into Extensible 3D (X3D)—a XML-based description language for 3D models (ISO/IEC 19775-1 2013). The actual XProc implementation of the pipeline is listed in Appendix B.2.2 on pages 196 f. The specific processing steps are described in the following. As in the preceding subgenre examples, first, the RST description of the given document is validated against the general fieldguide schema. Next, the RST description is transformed to X3D by means of XSLT templates that define the output structure and the descending nodes' content mode for each key structure R1–R4. The stylesheet implementation that creates the 3D-specific output is listed in Appendix B.2.2 on pages 197 ff. and specifies the following transformations:

- The nucleus specified by R1 that represents the major subject of the fieldguide entry is output as the heading of the first poster panel. As the panel is composed of five `x3d:Box` elements where one element describes the poster canvas and the others the four frame pieces, the poster contents are realized as `x3d:ImageTexture` for the canvas. In order to create the canvas texture directly from the XSLT stylesheet, the texture is described by means of the XML-based SVG that allows to embed the poster content via HTML (see McCormack et al. 2011, Sec. 23). Therefore, the nucleus is transformed into a `html:h1` element, inserted to the texture's SVG description via `svg:foreignObject`, and realized by means of the "list" mode.
- The satellites specified by R4 represent the classification of the document's subject and are output below the heading on the first poster panel. Identically to the print-specific output the classification satellites are emphasized via `html:em`, separated via `html:br`, and realized by means of the "list" mode.



(a) (b)

Figure 5.9.: *Seagull model*[11] *(a) with its underlying mesh and (b) as rendered within a web browser by means of the X3D Document Object Model (X3DOM) library (Behr et al. 2009)*

The core structure specified by R3 distinguishes the following three items:

- The first item represents the basic description of the fieldguide's subject and is output on the first poster panel. As the content of the poster should be quickly understood, the basic description is output in list form. Therefore, the first item is transformed into a `html:ul` element with its content nodes wrapped into `html:li` elements and realized by means of the "list" mode (as we suggest this mode as typical for poster presentations).
- The second item represents the portrayal of the fieldguide's subject and is output on the central stand of the scene. Being output in a three-dimensional context, the second item is added directly to the `x3d:Scene` element by means of the "3D" mode.
- The third item represents the supplementary fact-file of the fieldguide entry that is output on the second poster panel. Following the main mode of the posters, the third item is transformed into a `html:ul` element with its content nodes wrapped into `html:li` elements and realized by means of the "list" mode.

As this publishing channel uses a 3D representation of the document's subject, the content base has to provide a new content realization for the corresponding basic unit described by means of the "3D" mode. Figure 5.9 above shows a rendering of the added model[12]

11. Source: JB. *Updated Seagull.* 3D Warehouse, March 2014. `https://3dwarehouse.sketchup.com/model.html?id=1bf3d8226eba84df1d143af0a655075d` (05.03.2015).
12. Note, that the model represents a seagull instead of a gannet, as a corresponding gannet model was not freely available.

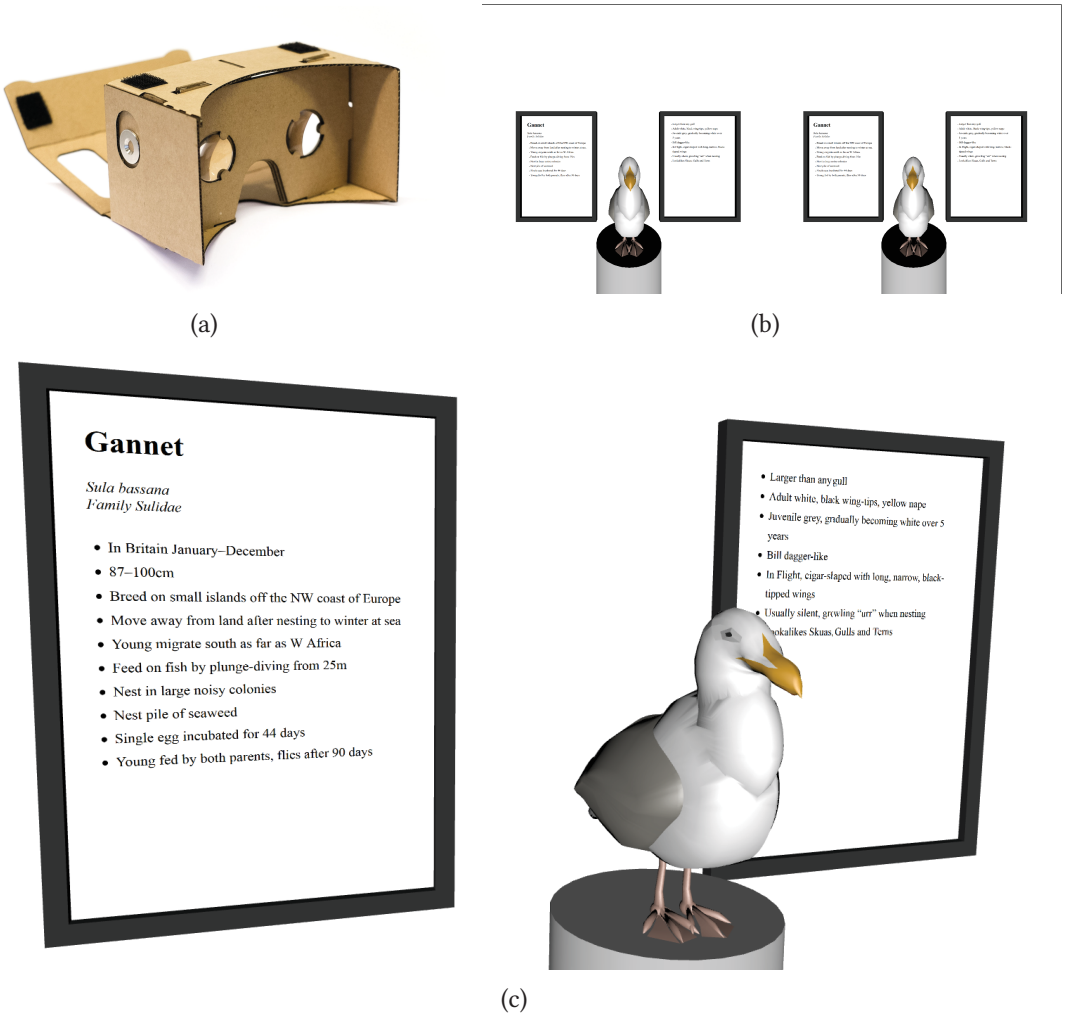


Figure 5.10.: *VR representation of the example document via (a) head-mounted display and (b) stereoscopic output that create a three-dimensional impression of (c) the output scene*

that serves as the new 3D realization and that is described by means of X3D as listed in Appendix B.2.1 on page 173. As the model description is embedded within HTML5, the `ID` attribute type has to be re-declared for the attributes that are used as fragment identifiers within the content base. Therefore, similarly to the movie-specific subgenre example, the following attribute-list is added to the abbreviated HTML5 document type declaration: `<!DOCTYPE html [<!ATTLIST Group DEF ID #IMPLIED>]>`.

The resulting 3D output from the content transformation of the example fieldguide is listed in Appendix B.2.2 on pages 204 ff. A corresponding output as rendered by a web browser within a Virtual Reality (VR) use case is shown in Figure 5.10 above.

5.3. Discussion

The preceding sections have provided an example application that describes a multistructured document and its channel-specific alternative structure transformations. Specifically, the example describes the transformation of the document's content structure into very diverging document representations (print, movie, and 3D). As demonstrated in the successive description of different publishing channels, new output transformations might need new content realizations. But it can be assumed that there is only a limited number of content modes to be expected in common document types. That is, new emerging output devices do not necessarily create new content modes, but rather use already existing modalities in a different way, e.g. by combining content modes that were strictly separated before. Suchlike documents would not constitute entirely new genres, but so-called hybrid genres. As an example, e-books mainly use the same modalities as their printed counterparts, but offer the integration possibilities of video, audio, and interactive content as well. The example document in this chapter comprises potential hybrid-subgenres already. As an example, a combination of the print layout and the movie output could be used for an e-book derivative. Conversely, new output devices could rely on a subset of an already existing publishing channel. As an example, a smart watch, being too small to efficiently display video content, could use the audio realization of the movie output. Moreover, the new document model allows to describe resources that are partially exceeding the automated document production. Rethinking Briet's example of a zoo (see Chap. 2), the 3D channel example could reference to an actual gannet, where the fieldguide information would be realized on an information plate (that can be created with the framework) or, in a prospective use case, the information would be displayed via Augmented Reality (AR) overlay.

6. Final Conclusions and Outlook

This thesis has introduced a revision of single source multichannel publishing workflows by elaborating a new underlying document model that comprises a media-independent content description logic and its transfer to media-specific representations. The proposed document model and the resulting publishing framework are based on informal studies that investigate the concept of genre. According to these studies a document is based on distinguishable but complementary characteristics that take not only the layout but additionally the rhetoric and the navigation of a document into account. Specific patterns between and within these interdependent characteristics constitute different genres that distinguish different types of documents. According to this model, publishing a book and publishing a website, for example, is not a transition between two distinct output media, but between two distinct genres (see Chap. 2).

The theoretical framework was formalized by extending the formal model of multistructured documents by the concept of genre. Specifically, a multistructured document describes alternative document structures and alternative content realizations of a concrete document by means of a single graph. A genre is thereby represented via functions that describe the mapping between alternative document structures. Thus, a specific genre is defined by typical characteristics within a (primary) document structure and how these characteristics are translated into other (secondary) document structures, where alternative mappings constitute alternative subgenres. More specific, the typical rhetorical structure for a given document type specifies the generic characteristics of a superordinated genre; the different output-specific realizations are described as subgenre-specific mappings (see Chap. 3).

Based on this formal model of genre-aware documents, a technical framework was implemented via XML-compliant languages and processes that realize a multichannel publishing workflow. The framework describes the underlying graph of a multistructured document via XLink and incorporated RDF statements that use RDF Schema and XML Schema to specify

the arcs' and nodes' semantics and that use DTD identifiers (e.g. of XHTML modules) to distinguish different content modes. As the former languages provide a generic approach to describe document structures, the framework is not depending on a specific document type but can represent any XML-based document description language (e.g. to describe the layout, rhetoric, and navigation of a document). A specific genre is described via Schematron to define the generic composition of the primary structure and via XSLT to specify the structure transformations. As all workflow steps rely on common XML standards the whole framework was realized on the basis of existing XML processing tools. Therefore, the complete publishing workflow itself was describable by means of XProc pipelines and successfully embedded in a common publishing environment (see Chap. 4).

Thus, these three consecutive document description approaches (the informal, formal, and technical document modeling) motivate a general shift of focus from media to genre in order to conceptualize automated multichannel publishing solutions. The functionality of the conceptualized framework was demonstrated by a running example (in Chap. 3–4) and in a more complex sample application (see Chap. 5).

Possible follow-up studies could start with the following questions.

In the sample application three specific languages were suggested as examples to describe the complementary document structures: HTML and associated languages to describe the layout, RST to describe the rhetoric, and NCX to describe the navigational anchor points of a document. While examples of the former two characteristics and languages were introduced, no particular use-case was discussed that describes the navigational or social aspect of a document. The main problem is the lack of a well-established navigation description language that is media-independent. In contrast, HTML provides a layout-oriented markup language that is used to describe documents of diverging output media—amongst others to describe web pages (Jacobs et al. 1999), mobile apps (Berjon et al. 2014), electronic books (Gylling et al. 2011), and printed matter (McCarron 2010, Kleinfeld 2013). Thus, HTML can be considered a general cross-media document description language. Similarly and even more generic, RST provides a media-independent language to describe the logical content structure of a document (Bateman 2008). But conversely, NCX is specialized in describing navigation information for paginated book-like documents (ANSI/NISO Z39.86-2005 (R2012)). The investigation of new description approaches for the navigational document aspects and their application to publishing workflows is a topic all its own; the integration of corresponding description solutions (e.g. Mons and Velterop 2009, Shotton 2009) might enhance and complete the introduced model of genre-aware documents.

Furthermore, within this thesis the content modes are mainly differentiated by means of XHTML modules and HTML-associated languages. This approach is especially appropriate where the output structures are based on XHTML as well. Other publishing contexts might differentiate between content modes by other means. As an example, for text-based documents TEI provides description modules that differentiate amongst others between verse texts, performance texts, and speech transcriptions (Burnard and Bauman 2015). Other document types might need to further subdivide graphic-based content modes (e.g. McCloud 1993, Chap. 2). Therefore, further studies on documents that are based on other main-modalities than the sample application of this thesis might lead to a general classification of existing types of content modes. As an example, sample documents might be graphic-based communications like comic strips or spatial communications like construction guidance. As both the technical implementation and its underlying models are not restricted to specific document description languages or modalities, the introduced framework is already prepared for these follow-up studies and use-cases.

FINIS.

Bibliography

Rocio Abascal, Michel Beigbeder, Aurélien Bénel, Sylvie Calabretto, Bertrand Chabbat, Pierre-Antoine Champin, Noureddine Chatti, David Jouve, Yannick Prié, Béatrice Rumpler, and Éric Thivant. Documents à structures multiples. In *Proceedings of the 2004 IEEE International Conference on Sciences of Electronic, Technologies of Information and Telecommunications (SETIT 2004)*, Sousse, March 2004. 19, 22, 30, 126

James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983. 43

Patrick Allen, John A. Bateman, and Judy Delin. Genre and layout design in multimodal documents — towards an empirical account. In *Proceedings of the American Association for Artificial Intelligence Fall Symposium on Using Layout for the Generation, Analysis, or Retrieval of Documents*, North Falmouth, Massachusetts, November 1999. 93

Jacques André, Richard Furuta, and Vincent Quint, editors. *Structured Documents*. Number 2 in Cambridge Series on Electronic Publishing. Cambridge University Press, New York, March 1989. 8, 122

ANSI/NISO Z39.86-2005 (R2012). Specifications for the digital talking book. Standard, National Information Standards Organization, Baltimore, April 2012. 76, 113

Daniel Austin, Shane McCarron, Subramanian Peruvemba, Masayasu Ishikawa, and Mark Birbeck. XHTML™ modularization 1.1 — second edition. W3C recommendation, W3C, July 2010. `http://www.w3.org/TR/2010/REC-xhtml-modularization-20100729/`. 48, 76, 97

Jon Barwise and John Perry. *Situations and Attitudes*. Bradford Book. MIT Press, Cambridge, January 1983. 16

John A. Bateman. *Multimodality and Genre — A Foundation for the Systematic Analysis of Multimodal Documents*. Palgrave Macmillan, London, April 2008. 4, 7, 14, 15, 77, 113

John A. Bateman. The decomposability of semiotic modes. In Kay O'Halloran and Bradley Smith, editors, *Multimodal Studies — Exploring Issues and Domains*, Routledge Studies in Multimodality, chapter 2, pages 17–38. Routledge, New York, June 2011. 13, 14, 15

John A. Bateman and Karl-Heinrich Schmidt. *Multimodal Film Analysis — How Films Mean*. Routledge Studies in Multimodality. Routledge, New York, September 2011. 13

Johannes Behr, Peter Eschler, Yvonne Jung, and Michael Zöllner. X3DOM: A DOM-based HTML5/X3D integration model. In *Proceedings of the 14th International Conference on 3D Web Technology (Web3D '09)*, pages 127–135, Darmstadt, Germany, June 2009. 109

Anders Berglund. Extensible stylesheet language (XSL) version 1.1. W3C recommendation, W3C, December 2006. `http://www.w3.org/TR/2006/REC-xsl11-20061205/`. 1

Robin Berjon, Steve Faulkner, Travis Leithead, Erika Doyle Navara, Edward O'Connor, Silvia Pfeiffer, and Ian Hickson. HTML 5. W3C recommendation, W3C, October 2014. `https://www.w3.org/TR/2014/REC-html5-20141028/`. 48, 105, 113

Steven Bird and Mark Liberman. A formal framework for linguistic annotation. *Speech Communication*, 33(1–2):23–60, January 2001. 18

Paul V. Biron and Ashok Malhotra. XML schema part 2: Datatypes second edition. W3C recommendation, W3C, October 2004. `http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/`. 46, 131

Tim Bray, François Yergeau, C. M. Sperberg-McQueen, Jean Paoli, and Eve Maler. Extensible markup language (XML) 1.0 (fourth edition). W3C recommendation, W3C, August 2006. `http://www.w3.org/TR/2006/REC-xml-20060816/`. 1

Suzanne Briet. *Qu'est-ce que la documentation ?* Éditions Documentaires Industrielles et Techniques (ÉDIT), Paris, 1951. 6

Michael Keeble Buckland. What is a "document"? *Journal of the American Society of Information Science*, 48(9):804–809, September 1997. 7

Lou Burnard and Syd Bauman. TEI P5: Guidelines for electronic text encoding and interchange. 2.8.0. 06-apr-2015. Technical report, TEI Consortium, April 2015. `http://www.tei-c.org/Guidelines/P5/`. 3, 43, 44, 114

Gavin Carothers and Eric Prud'hommeaux. RDF 1.1 turtle. W3C recommendation, W3C, February 2014. `http://www.w3.org/TR/2014/REC-turtle-20140225/`. 46

Noureddine Chatti, Suha Kaouk, Sylvie Calabretto, and Jean Marie Pinon. MultiX: an XML based formalism to encode multi-structured documents. In *Proceedings of Extreme Markup Languages*, Montréal Québec, 2007. 44, 74

Ron Daniel. Harvesting RDF statements from XLinks. W3C note, W3C, September 2000. `http://www.w3.org/TR/2000/NOTE-xlink2rdf-20000929`. 63

Ronald E. Day, Laurent Martinet, and Hermina G. B. Anghelescu. *What is Documentation? — English Translation of the Classic French Text*. Scarecrow Press, Lanham, March 2006. 6

Steven J. DeRose. Markup overlap: A review and a horse. In *Proceedings of Extreme Markup Languages*, Montréal Québec, 2004. 42

Steven J. DeRose, David G. Durand, Elli Mylonas, and Allen H. Renear. What is text, really? *Journal of Computing in Higher Education*, 1(2):3–26, 1990. 3

Keith J. Devlin. *Logic and Information.* Cambridge University Press, Cambridge, August 1990. 16

Keith J. Devlin and Duska Rosenberg. *Language at work — analyzing communication breakdown in the workplace to inform systems design.* Number 66 in CSLI Lecture Notes. CSLI Publications, Stanford, June 1996. 16

Keith J. Devlin and Duska Rosenberg. Information in the study of human interaction. In Pieter Adriaans and Johan van Benthem, editors, *Philosophy of Information*, number 8 in Handbook of the Philosophy of Science, chapter E, pages 685–710. North-Holland, Amsterdam, The Netherlands, May 2006. 16

DIN 66261. Information processing — nassi-shneiderman flowchart symbols. Standard, Deutsches Institut für Normung e. V., Berlin, November 1985. 64

Stefanie Dipper and Michael Götze. Accessing heterogeneous linguistic data — generic XML-based representation and flexible visualization. In *Proceedings of the 2nd Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 206–210, Poznan, April 2005. 44

Patrick Durusau and Matthew Brook O'Donnell. Concurrent markup for XML documents. In *Proceedings of XML Europe 2002*, Barcelona, May 2002. 42, 43

Rainer Eckstein and Silke Eckstein. *XML und Datenmodellierung — XML-Schema und RDF zur Modellierung von Daten und Metadaten einsetzen.* xml.bibliothek. dpunkt.verlag, Heidelberg, 2004. 62

Ramanathan Guha and Dan Brickley. RDF schema 1.1. W3C recommendation, W3C, February 2014. `http://www.w3.org/TR/2014/REC-rdf-schema-20140225/`. 46

Markus Gylling, William McCoy, Elika J. Etemad, and Matt Garrish. EPUB content documents 3.0. IDPF recommended specification, IDPF, October 2011. `http://www.idpf.org/epub/30/spec/epub30-contentdocs-20111011.html`. 113

Renate Henschel, John Bateman, and Judy Delin. Automatic genre-driven layout generation. In *Proceedings of the 6th "Konferenz zur Verarbeitung natürlicher Sprache" (KONVENS) Conference*, Saarbrücken, September 2002. 93, 97, 99, 102, 138

Peter Holden. *Birds of Britain and Ireland.* Collins Wild Guide. Harper Collins, London, April 1996. 94, 103

Nancy Ide, Patrice Bonhomme, and Laurent Romary. XCES: An XML-based encoding standard for linguistic corpora. In *Proceedings of the Second International Language Resources and Evaluation Conference (LREC)*, Athens, Greece, June 2000. 44

Masayasu Ishikawa and Shane McCarron. XHTML™ 1.1 — module-based XHTML — second edition. W3C recommendation, W3C, November 2010. `http://www.w3.org/TR/2010/REC-xhtml11-20101123/`. 133

ISO 24612. Language resource management — linguistic annotation framework (LAF). Standard, International Organization for Standardization, Geneva, June 2012. 44

ISO 5127. Information and documentation — vocabulary. Standard, International Organization for Standardization, Geneva, October 2001. 5

ISO 639-1. Codes for the representation of names of languages — part 1: Alpha-2 code. Standard, International Organization for Standardization, Geneva, 2002. 25

ISO 8879. Information processing — text and office systems — standard generalized markup language (SGML). Standard, International Organization for Standardization, Geneva, October 1986. 43

ISO/IEC 19757-3. Information technology — document schema definition language (DSDL) — part 3: Rule-based validation — schematron. Standard, International Organization for Standardization, Geneva, June 2006. 79

ISO/IEC 19775-1. Information technology — computer graphics, image processing and environmental data representation — extensible 3D (X3D) — part 1: Architecture and base components. Standard, International Organization for Standardization, Geneva, November 2013. 108

ISO/IEC 8613-2. Information technology — open document architecture (ODA) and interchange format — document structures. Standard, International Telecommunication Union (ITU), Helsinki, March 1993. (ITU-T Recommendation T.412). 7

Ian Jacobs, David Raggett, and Arnaud Le Hors. HTML 4.01 specification. W3C recommendation, W3C, December 1999. `http://www.w3.org/TR/1999/REC-html401-19991224/`. 113

Michael Kay. XSL transformations (XSLT) version 2.0. W3C recommendation, W3C, January 2007. `http://www.w3.org/TR/2007/REC-xslt20-20070123/`. 64, 76, 82

Sanders Kleinfeld. The case for authoring and producing books in (X)HTML5. In *Proceedings of Balisage: The Markup Conference 2013*, volume 10 of *Balisage Series on Markup Technologies*, Montréal, August 2013. 113

Donald Ervin Knuth. *The TEXbook*, volume A of *Computers & Typesetting*. Addison-Wesley, March 1986. 104

Mounia Lalmas. The flow of information in information retrieval — towards a general framework for the modelling of information retrieval. In Fabio Crestani, Mounia Lalmas, and Cornelis Joost van Rijsbergen, editors, *Information Retrieval — Uncertainty and Logics — Advanced Models for the Representation and Retrieval of Information*, The Kluwer International Series on Information Retrieval, chapter 6, pages 129–150. Springer US, Glasgow, 1998. 16

Bing Liu, editor. *Web Data Mining — Exploring Hyperlinks, Contents, and Usage Data*. Data-Centric Systems and Applications. Springer, Chicago, 2007. 17

Niels Windfeld Lund. Documentation in a complementary perspective. In W. Boyd Rayward, editor, *Aware and Responsible — Papers of the Nordic-International Colloquium on Social and Cultural Awareness and Responsibility in Library, Information and Documentation Studies (SCARLID)*, chapter 5, pages 93–102. Scarecrow Press, Lanham, December 2003a. 12, 15

Niels Windfeld Lund. Doceo + mentum — a ground for a new discipline. In *Proceedings of the 1st Document Research Conference (DOCAM '03)*, UC Berkeley, August 2003b. 12, 15

Niels Windfeld Lund and Roswitha Skare. Document theory. In Marcia J. Bates and Mary Niles Maack, editors, *Encyclopedia of Library and Information Sciences*, chapter 162, pages 1632–1639. Taylor & Francis, Tromsø, third edition, December 2009. 5, 7

Eve Maler, Norman Walsh, David Orchard, and Steven DeRose. XML linking language (XLink) version 1.1. W3C recommendation, W3C, May 2010. `http://www.w3.org/TR/2010/REC-xlink11-20100506/`. 44, 45, 64, 66, 133

Murray Maloney, David Beech, Henry Thompson, and Noah Mendelsohn. XML schema part 1: Structures second edition. W3C recommendation, W3C, October 2004. `http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/`. 46, 131

William C. Mann and Maite Taboada. Rhetorical structure theory — relation definitions. `http://www.sfu.ca/rst/01intro/definitions.html` (26.11.2015), 2005. 78, 134

William C. Mann and Sandra Annear Thompson. Rhetorical structure theory — toward a functional theory of text organization. *Text*, 8(3):243–281, 1988. 14, 77, 78

Benoît Marchal. *XML by Example*. By Example Series. Que Publishing, Indianapolis, second edition, 2002. 47

Yves Marcoux, Michael Sperberg-McQueen, and Claus Huitfeldt. Modeling overlapping structures — graphs and serializability. In *Proceedings of Balisage: The Markup Conference 2013*, volume 10 of *Balisage Series on Markup Technologies*, Montréal, Canada, August 2013. 18

Jonathan Marsh, Eve Maler, Norman Walsh, and Paul Grosso. XPointer framework. W3C recommendation, W3C, March 2003. `http://www.w3.org/TR/2003/REC-xptr-framework-20030325/`. 44

Jonathan Marsh, Daniel Veillard, and David Orchard. XML inclusions (XInclude) version 1.0 (second edition). W3C recommendation, W3C, November 2006. `http://www.w3.org/TR/2006/REC-xinclude-20061115/`. 44

Shane McCarron. XHTML-print — second edition. W3C recommendation, W3C, November 2010. `http://www.w3.org/TR/2010/REC-xhtml-print-20101123/`. 113

Scott McCloud. *Understanding Comics — The Invisible Art.* Tundra Publishing, Northampton, Massachusetts, 1993. 114

Cameron McCormack, Jonathan Watt, Doug Schepers, Anthony Grasso, Patrick Dengler, Jon Ferraiolo, Erik Dahlström, Dean Jackson, Jun Fujisawa, and Chris Lilley. Scalable vector graphics (SVG) 1.1 (second edition). W3C recommendation, W3C, August 2011. `http://www.w3.org/TR/2011/REC-SVG11-20110816/`. 108

Wiebke Möhr and Ingrid Schmidt, editors. *SGML und XML — Anwendungen und Perspektiven.* Springer-Verlag, Berlin, Heidelberg, 1999. 1

Barend Mons and Jan Velterop. Nano-publication in the e-science era. In *Proceedings of the Workshop on Semantic Web Applications in Scientific Discourse (SWASD 2009)*, Washington DC, USA, October 2009. 113

Michael O'Donnell. RSTTool 2.4 — a markup tool for rhetorical structure theory. In *Proceedings of the International Natural Language Generation Conference (INLG'2000)*, pages 253–256, Mitzpe Ramon, June 2000. 76, 78, 79, 80, 224

Paul Otlet. *Traité de Documentation — Le Livre sur le Livre — Théorie et Pratique.* Editiones Mundaneum, Palais Mondial, Bruxelles, 1934. 5, 6

Tobias Ott. *Crossmediales Publizieren im Verlag.* Walter de Gruyter GmbH, Berlin/Boston, December 2013. 1, 8

Roger T. Pédauque. Document: Form, sign and medium, as reformulated for electronic documents. `http://archivesic.ccsd.cnrs.fr/docs/00/06/22/28/PDF/sic_00000594.pdf` (30.12.2012), July 2003. 9, 10, 11

Roger T. Pédauque. *Le document à la lumière du numérique.* C&F éditions, Caen, September 2006a. 4, 8, 15, 120

Roger T. Pédauque. Document: forme, signe et médium, les re-formulations du numérique. In *Le document à la lumière du numérique* Pédauque (2006a), chapter "Pédauque 1", pages 27–78. 9, 10, 11

Roger T. Pédauque. Le texte en jeu, permanence et transformations du document. In *Le document à la lumière du numérique* Pédauque (2006a), chapter "Pédauque 2", pages 83–155. 62

Roger T. Pédauque. *La redocumentarisation du monde.* Cépaduès éditions, Toulouse, January 2007. 4, 8, 12, 15

John Plaice and Chris Rowley. Characters are not simply names, nor documents trees. In *Glyph and Typesetting Workshop*, pages 9–16, Kyoto, Japan, 2003. East Asian Center for Informatics in Humanities, the 21st Century COE, Kyoto University. 18

Pierre-Edouard Portier, Noureddine Chatti, Sylvie Calabretto, Elöd Egyed-Zsigmond, and Jean-Marie Pinon. Modeling, encoding and querying multi-structured documents. *Information Processing & Management*, 48(5):931–955, September 2012. 19, 21, 22, 30, 128

Jean-Michel Salaün. *Vu, lu, su — Les architectes de l'information face à l'oligopole du Web.* La Découverte, Paris, March 2012. 10, 11

Jean-Michel Salaün and Collegium de Lyon. Une approche documentaire du web. `http://www.youtube.com/watch?v=5ICyFJouHv4` (27.11.2011), June 2011. 10, 11

Kurt Sandkuhl. First steps to cross media publishing and multimodal documents. In *Principles of Document Processing — Third International Workshop (PODP'96)*, pages 15–26, Palo Alto, California, USA, September 1996. 1

Frederik R. N. Schlupkothen. HTML to LaTeX transformation. *TUGboat — The Communications of the TeX Users Group*, 35(1):83–90, April 2014. 104

Frederik R. N. Schlupkothen and Karl-Heinrich Schmidt. A genre-aware document model for multichannel publishing workflows. In *Proceedings of the Fourteenth International Symposium on Spatial Media (ISSM'13–'14)*, Aizu-Wakamatsu, Japan, February 2014. 5

David Shotton. Semantic publishing — the coming revolution in scientific journal publishing. *Learned Publishing*, 22(2):85–94, April 2009. 113

C. M. Sperberg-McQueen and Claus Huitfeldt. GODDAG: A data structure for overlapping hierarchies. In Peter King and Ethan V. Munson, editors, *Digital Documents: Systems and Principles*, Lecture Notes in Computer Science, pages 139–160. Springer Berlin Heidelberg, 2004. 42

Maik Stührenberg and Daniel Jettka. A toolkit for multi-dimensional markup — the development of SGF to XStandoff. In *Proceedings of Balisage: The Markup Conference 2009*, volume 3 of *Balisage Series on Markup Technologies*, Montréal, Canada, August 2009. 44

John Malcolm Swales. *Genre Analysis — English in Academic and Research Settings.* Cambridge Applied Linguistics Series. Cambridge University Press, Ann Arbor, November 1990. 3, 125

John Malcolm Swales. *Research Genres — Explorations and Applications.* Cambridge Applied Linguistics Series. Cambridge University Press, Cambridge, November 2004. 3

Y. Y. Tang, C. Y. Suen, C. D. Yan, and M. Cheriet. Document analysis and understanding — a brief survey. In *Proceedings of the 1st International Conference on Document Analysis and Recognition (ICDAR '91)*, Saint-Malo, September 1991. 17

Henry Thompson, Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin. Namespaces in XML 1.0 (third edition). W3C recommendation, W3C, December 2009. `http://www.w3.org/TR/2009/REC-xml-names-20091208/`. 51, 71

Henry S. Thompson and David McKelvie. Hyperlink semantics for standoff markup of read-only documents. In *Proceedings of SGML Europe '97: The next decade — Pushing the Envelope*, page 227–229, Barcelona, May 1997. 43

Henry S. Thompson, Norman Walsh, and Alex Milowski. XProc: An XML pipeline language. W3C recommendation, W3C, May 2010. `http://www.w3.org/TR/2010/REC-xproc-20100511/`. 64, 74

Richard Tobin. An RDF schema for the XML information set. W3C note, W3C, April 2001. `http://www.w3.org/TR/2001/NOTE-xml-infoset-rdfs-20010406`. 63

Jacques Virbel. The contribution of linguistic knowledge to the interpretation of text structures. In André et al. (1989), chapter 9, pages 161–180. 7

David Wood, Markus Lanthaler, and Richard Cyganiak. RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C, February 2014. `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/`. 46

Appendixes

A. Models, Concepts, Source Material

This appendix compiles background information like the theoretical resources and practical source material of the thesis.

Specifically, Appendix A.1 provides the original description of the notion of genre (Chap. 1); Appendix A.2 provides the original formalization for the introduced genre-aware document model (Chap. 2); Appendix A.3 provides the list of XML Schema fragment identifiers that are used as node and arc types in the technical implementation (Chap. 4); Appendix A.4 provides the DTD customization that adds the XLink attributes to XHTML as used in the multistructured document examples (Chap. 4, 5); Appendix A.5 provides the classic RST relation definitions that are used to describe a document's logical content structure (Chap. 4, 5); finally, Appendix A.6 provides the original source material of the "Gannet" example document (Chap. 5).

A.1. Working Description of Genre

Genre as described by John Swales (excerpts from Swales 1990, pp. 24–27, 45–58).

"A genre comprises a class of communicative events, the members of which share some sets of communicative purposes. These purposes are recognized by the expert members of the parent discourse community [. . .]. Communicative purpose is both a privileged criterion and one that operates to keep the scope of a genre as here conceived narrowly focused on comparable rhetorical action. In addition to purpose, exemplars of a genre exhibit various patterns of similarity in terms of structure, style, content and intended audience. If all high probability expectations are realized, the exemplar will be viewed as prototypical by the parent discourse community. The genre names inherited and produced by discourse communities and imported by others constitute valuable ethnographic communication, but typically need further validation." (Swales 1990, p. 58)

Accordingly, Swales (1990, pp. 24–27) conceptualizes a discourse community by means of the following headlines:

1. A discourse community has a broadly agreed set of common public goals.
2. A discourse community has mechanisms of intercommunication among its members.
3. A discourse community uses its participatory mechanisms primarily to provide information and feedback.
4. A discourse community utilizes and hence possesses one or more genres in the communicative furtherance of its aims.
5. In addition to owning genres, a discourse community has acquired some specific lexis.
6. A discourse community has a threshold level of members with a suitable degree of relevant content and discoursal expertise.

Likewise, Swales (1990, pp. 45–57) provides a working definition of genre:

1. A genre is a class of communicative events.
2. The principal criterial feature that turns a collection of communicative events into a genre is some shared set of communicative purpose.
3. Exemplars or instances of genres vary in their prototypicality.
4. The rationale behind a genre establishes constraints on allowable contributions in terms of their content, positioning and form.
5. A discourse community's nomenclature for genres is an important source of insight.

A.2. Formal Document Models

Original formal definition of multistructured documents (Sec. A.2.1) and its revision (Sec. A.2.2).

A.2.1. Multistructured Document Model (Abascal et al.)

Core-definitions of multistructured documents (Abascal et al. 2004, translated by F. S.).

Definition A.1 (Documental Structure). A *documental structure* is a description of a document by a set of elements that are in relation to each other in the course of or with regard to a usage. Mathematically: There is a set of elements and binary relations. Therefore a documental structure is a labeled multigraph

$$S = \langle N, L_N, l_N, L_A, A \rangle$$

where:

- N is the set of nodes of the graph (elements of the structure)
- L_N is the set of labels of the nodes ("content" of elements)
- $l_N : N \to L_N$ is the function that associates to each node its label
- L_A is the set of labels of the arcs ("names" of relations)
- $A \subseteq N \times N \times L_A$ is the set of arcs (named relations between elements)
- we add as constraint that the graph shall be connected

Definition A.2 (Correspondence). A *correspondence* between two structures is a non-empty binary relation from the elements of the first towards the elements of the second structure.

We note $\mathrm{corr}(S_i, S_j)$ the set of possible correspondences between two structures S_i and S_j. Formally

$$\mathrm{corr}\left(S_i, S_j\right) = \wp\left(N_i \times N_j\right) - \{\varnothing\}$$

where $\wp(E)$ designates the set of subsets of a set E and $S_i = \langle N_i, L_{N_i}, l_{N_i}, L_{A_i}, A_i \rangle$ and $S_j = \langle N_j, L_{N_j}, l_{N_j}, L_{A_j}, A_j \rangle$.

Definition A.3 (Multistructured Document). A *multistructured document* is a structured document in which we consider multiple possible usages and therefore multiple structural decompositions. One of these structures, called *primary structure*, is constitutive of the document in terms of unit. Any other structure relies on the primary structure by means of a *correspondence* with it, directly or by way of another structure. Finally, two arbitrary structures cannot be in mutual correspondence, neither directly nor indirectly. Formally

$$D = \langle S_0, \Sigma, C \rangle$$

where:

- S_0 is the primary structure of the document,
- $\Sigma = \{S_i \mid i = 1..n\}$ is the set of the other structures of the document,
- $C = \left\{C_{ij} \in \text{corr}\left(S_i, S_j\right)\right\}$ is the set of correspondences so that:
  - $\forall i \neq 0, \exists j < i \mid C_{ij} \in C$ (connectivity and convergence),
  - $\forall i \leq j, C_{ij} \notin C$ (absence of cycles).

Firstly we note that a monostructured document is a particular multistructured document $D = \langle S_0, \varnothing, \varnothing \rangle$. Then again the set C of correspondences between structures defines a relation between structures that permits to consider a *graph of structures*. The constraints imposed to C confer certain properties on this graph:

- it is connected and possesses as its single sink S_0 (all correspondences "converge" to S_0), since all structure relies directly or indirectly on the primary structure,
- it comprises no cycle: effectively, if two structures are defined so as to mutually referencing each other, we consider them to constitute a single structure of usage.

Corpus and Catalog

Definition A.4 (Corpus). We call a set of multistructured documents *corpus*.

Definition A.5 (Documental Multistructure). A *documental multistructure M* is a set of documental structures put in correspondence. Formally

$$M = \langle \Sigma, C \rangle$$

where:

- $\Sigma = \{S_i \mid i = 1..n\}$ is the set of documental structures
- $C = \left\{C_{ij} \in \text{corr}\left(S_i, S_j\right)\right\}$ so that $\forall i \leq j, C_{ij} \notin C$ (absence of cycles)

We note that the definition of a *multistructured document* (Definition A.3) verifies the definition of a *documental multistructure* by adding the constraints imposing the "convergence" and the connectivity of the graph of structures. Furthermore, a set of multistructured documents (corpus) constitutes a documental multistructure.

Definition A.6 (Catalog). A *catalog* is a documental manifestation of a corpus. Being a document the catalog possesses a multistructure conforming to the definition of multistructured documents. Being a manifestation of a corpus, we can identify in its multistructure its constituting documents.

For example conference proceedings or a thesis containing figures (that can be seen as a thesis or as a manifestation of a corpus of figures that is used by the examinee) are catalogs.

A.2.2. Multistructured Document Model (Portier et al.)

Revised definitions of multistructured documents (Portier et al. 2012).

Definition A.7 (Multi-structured Document). A multi-structured document D is a graph defined by the triplet

$$D = \langle BS, \Sigma, C \rangle$$

where:

- BS is an oriented graph called the basic structure of D;
- Σ is a set of graphs called the documentary structures;
- C is a set of relations between nodes of Σ-graphs or BS. We call these relations correspondences between structures.

Notations

- $\mathrm{F}(D)$ designates the set of all possible content fragments of a document D.
- $\Phi(D) = \{\varphi\colon (\mathrm{F}(D))^p \to \mathrm{F}(D) \mid p \geq 1\}$ is the set of all the functions that compose a piece of content from a series of smaller pieces. For example, the function that creates the string "ab" by concatenation of the two fragments $(a, b) \in \mathrm{F}(D)^2$ is an element of $\Phi(D)$.

Definition A.8 (Basic Structure). The basic structure of a multi-structured document D is an internal structure that stands as a base for sharing the same content among several structures. It is formally defined as a graph

$$BS = (v, e)$$

where:

- *root* is the root node of *BS*, each node is part of a path originating from it. This node has no associated semantic; it only provides connectivity to the graph.
- $\mathrm{F}_{\mathrm{Base}}(D) \subset \mathrm{F}(D)$ is the set of all disjoint fragments, called base fragments, from which other fragments can be composed and associated to document structures through correspondences. Each node $n \in \mathrm{F}_{\mathrm{Base}}(D)$ is the destination of one or more edges from composition nodes.
- N_{comp} is the set of composition nodes, i.e. the nodes that represent parts of content composed of smaller fragments. Each node $n \in N_{\mathrm{comp}}$ is the destination of exactly one edge with origin the root node. So the subgraph of *BS* whose nodes are $\{root\} \cup N_{\mathrm{comp}}$ is a trivial tree. Moreover each node $n \in N_{\mathrm{comp}}$ is labeled with a function $f \in \Phi(D)$ that prescribes the composition of the elements of $\mathrm{F}_{\mathrm{Base}}(D)$ linked to n.
- $v = \{root\} \cup N_{\mathrm{comp}} \cup \mathrm{F}_{\mathrm{Base}}(D)$ is the set of vertices of *BS*.
- $\mathrm{comp}(n)$ with $n \in N_{\mathrm{comp}}$ is the set of all the edges from the composition node n to one or more elements of $\mathrm{F}_{\mathrm{Base}}(D)$. Each of these edges is labeled with an integer, that makes $\mathrm{comp}(n)$ a totally ordered set. This order may be used by a function $f \in \Phi(D)$ associated with n to compose the elements of $\mathrm{F}_{\mathrm{Base}}(D)$ in the appropriate order.
- $e = \left\{(root, c) \mid c \in N_{\mathrm{comp}}\right\} \cup \left(\bigcup_{n \in N_{\mathrm{comp}}} \mathrm{comp}(n)\right)$ is the set of edges of *BS*.

Definition A.9 (Documentary Structure). A documentary structure is a set of structured descriptors applied to documentary content. Formally, a documentary structure is a tree defined by:

$$DS = \langle E, A, L, l_E, l_A \rangle$$

where:

- E is a finite set of vertices we call *structural elements*.
- $A \subset E \times E$ is a finite set of binary relations between structural elements. These relations must form a tree.

- L is a finite set of labels.
- $l_E : E \to L$ is a function that associates each structural element in E with a label in L.
- $l_A : A \to L$ is a function that associates each edge in A with a label in L.

A.3. XML Schema 1.0 Fragment Identifiers

All 130 fragment identifiers from the normative XML Schema 1.0 schema definition (Maloney et al. 2004, Biron and Malhotra 2004, Appx. A) in order of their occurrence within the schema implementation (Sec. A.3.1) and in alphabetic order (Sec. A.3.2). Emphasized fragment identifiers are built-in data types as described by Biron and Malhotra (2004).

A.3.1. Contextual Order

XML Schema fragment identifiers in order of their implementation occurrence.

#schema, #anyAttribute, #complexContent, #simpleContent, #complexType, #element, #all, #choice, #sequence, #group, #any, #attribute, #attributeGroup, #include, #redefine, #import, #selector, #field, #unique, #key, #keyref, #notation, #appinfo, #documentation, #annotation, **#string**, #string.preserve, **#boolean**, #boolean.whiteSpace, **#float**, #float.whiteSpace, **#double**, #double.whiteSpace, **#decimal**, #decimal.whiteSpace, **#duration**, #duration.whiteSpace, **#dateTime**, #dateTime.whiteSpace, **#time**, #time.whiteSpace, **#date**, #date.whiteSpace, **#gYearMonth**, #gYearMonth.whiteSpace, **#gYear**, #gYear.whiteSpace, **#gMonthDay**, #gMonthDay.whiteSpace, **#gDay**, #gDay.whiteSpace, **#gMonth**, #gMonth.whiteSpace, **#hexBinary**, #hexBinary.whiteSpace, **#base64Binary**, #base64Binary.whiteSpace, **#anyURI**, #anyURI.whiteSpace, **#QName**, #QName.whiteSpace, **#NOTATION**, #NOTATION.whiteSpace, **#normalizedString**, #normalizedString.whiteSpace, **#token**, #token.whiteSpace, **#language**, #language.pattern, **#IDREFS**, #IDREFS.minLength, **#ENTITIES**, #ENTITIES.minLength, **#NMTOKEN**, #NMTOKEN.pattern, **#NMTOKENS**, #NMTOKENS.minLength, **#Name**, #Name.pattern, **#NCName**, #NCName.pattern, **#ID**, **#IDREF**, **#ENTITY**, **#integer**, #integer.fractionDigits, **#nonPositiveInteger**, #nonPositiveInteger.maxInclusive, **#negativeInteger**, #negativeInteger.maxInclusive, **#long**, #long.minInclusive, #long.maxInclusive, **#int**, #int.minInclusive, #int.maxInclusive, **#short**, #short.minInclusive, #short.maxInclusive, **#byte**, #byte.minInclusive, #byte.maxInclusive, **#nonNegativeInteger**, #nonNegativeInteger.minInclusive, **#unsignedLong**, #unsignedLong.maxInclusive, **#unsignedInt**, #unsignedInt.maxInclusive, **#unsignedShort**, #unsignedShort.maxInclusive, **#unsignedByte**, #unsignedByte.maxInclusive, **#positiveInteger**, #positiveInteger.minInclusive, #simpleType, #restriction, #list, #union, #minExclusive, #minInclusive, #maxExclusive, #maxInclusive, #totalDigits, #fractionDigits, #length, #minLength, #maxLength, #enumeration, #whiteSpace, #pattern

A.3.2. Alphabetical Order

XML Schema fragment identifiers in alphabetical order.

#all, #annotation, #any, #anyAttribute, **#anyURI**, #anyURI.whiteSpace, #appinfo, #attribute, #attributeGroup, **#base64Binary**, #base64Binary.whiteSpace, **#boolean**, #boolean.whiteSpace, **#byte**, #byte.minInclusive, #byte.maxInclusive, #choice, #complexContent, #complexType, **#date**, #date.whiteSpace, **#dateTime**, #dateTime.whiteSpace, **#decimal**, #decimal.whiteSpace, #documentation, **#double**, #double.whiteSpace, **#duration**, #duration.whiteSpace, #element, **#ENTITIES**, #ENTITIES.minLength, **#ENTITY**, #enumeration, #field, **#float**, #float.whiteSpace, #fractionDigits, **#gDay**, #gDay.whiteSpace, **#gMonth**, #gMonth.whiteSpace, **#gMonthDay**, #gMonthDay.whiteSpace, #group, **#gYear**, #gYear.whiteSpace, **#gYearMonth**, #gYearMonth.whiteSpace, **#hexBinary**, #hexBinary.whiteSpace, **#ID**, **#IDREF**, **#IDREFS**, #IDREFS.minLength, #import, #include, **#int**, #int.minInclusive, #int.maxInclusive, **#integer**, #integer.fractionDigits, #key, #keyref, **#language**, #language.pattern, #length, #list, **#long**, #long.minInclusive, #long.maxInclusive, #maxExclusive, #maxInclusive, #maxLength, #minExclusive, #minInclusive, #minLength, **#Name**, #Name.pattern, **#NCName**, #NCName.pattern, **#negativeInteger**, #negativeInteger.maxInclusive, **#NMTOKEN**, #NMTOKEN.pattern, **#NMTOKENS**, #NMTOKENS.minLength, **#nonNegativeInteger**, #nonNegativeInteger.minInclusive, **#nonPositiveInteger**, #nonPositiveInteger.maxInclusive, **#normalizedString**, #normalizedString.whiteSpace, #notation, **#NOTATION**, #NOTATION.whiteSpace, #pattern, **#positiveInteger**, #positiveInteger.minInclusive, **#QName**, #QName.whiteSpace, #redefine, #restriction, #schema, #selector, #sequence, **#short**, #short.minInclusive, #short.maxInclusive, #simpleContent, #simpleType, **#string**, #string.preserve, **#time**, #time.whiteSpace, **#token**, #token.whiteSpace, #totalDigits, #union, #unique, **#unsignedByte**, #unsignedByte.maxInclusive, **#unsignedInt**, #unsignedInt.maxInclusive, **#unsignedLong**, #unsignedLong.maxInclusive, **#unsignedShort**, #unsignedShort.maxInclusive, #whiteSpace

A.4. XHTML plus XLink Document Type Definition

Module-based XHTML customization that adds XLink 1.1 attributes (Maler et al. 2010) to the XHTML 1.1 DTD (Ishikawa and McCarron 2010).

`xhtml-xlink.dtd`

```
<!-- .................................................................... -->
<!-- XHTML 1.1 plus XLink 1.1 DTD  ....................................... -->
<!-- URI: xhtml-xlink.dtd
     Author:     Frederik R.N. Schlupkothen <schlupko@uni-wuppertal.de>
-->
<!ENTITY % XHTML.version
         "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" >

<!ENTITY % XLINK.xmlns.attrib
         "xmlns:xlink CDATA #FIXED 'http://www.w3.org/1999/xlink'" >

<!-- add global XLink attributes (without parent-child syntax validation) -->
<!ENTITY % Core.extra.attrib
         "xml:base       CDATA #IMPLIED
          xlink:type    (simple|extended|locator|arc|resource|title) #IMPLIED
          xlink:href     CDATA #IMPLIED
          xlink:role     CDATA #IMPLIED
          xlink:arcrole  CDATA #IMPLIED
          xlink:title    CDATA #IMPLIED
          xlink:show    (new|replace|embed|other|none) #IMPLIED
          xlink:actuate (onLoad|onRequest|other|none) #IMPLIED
          xlink:label    NMTOKEN #IMPLIED
          xlink:from     NMTOKEN #IMPLIED
          xlink:to       NMTOKEN #IMPLIED" >

<!ENTITY % XHTML11.dtd
    PUBLIC "-//W3C//DTD XHTML 1.1//EN"
           "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" >
%XHTML11.dtd;

<!-- end of xhtml-xlink.dtd -->
```

A.5. RST Relation Definitions

Tables of classic RST relation definitions, organized by presentational (Sec. A.5.1), subject-matter (Sec. A.5.2), and multinuclear (Sec. A.5.3) relations (Mann and Taboada 2005), where relation types are distinguished by constraints on the nuclei (*N*) and satellites (*S*) and by the intended effect of the writer (*W*) on the reader (*R*).

A.5.1. Presentational Relations

Table A.1.: *Definitions of presentational relations*

Relation	Constraints on N or S	Intention of W
Antithesis	**N**: W has positive regard for N **N + S**: N and S are in contrast (see the Contrast relation); because of the incompatibility that arises from the contrast, one cannot have positive regard for both of those situations; comprehending S and the incompatibility between the situations increases R's positive regard for N	R's positive regard for N is increased
Background	**N**: R won't comprehend N sufficiently before reading text of S **N + S**: S increases the ability of R to comprehend an element in N	R's ability to comprehend N increases
Concession	**N**: W has positive regard for N **S**: W is not claiming that S does not hold; **N + S**: W acknowledges a potential or apparent incompatibility between N and S; recognizing the compatibility between N and S increases R's positive regard for N	R's positive regard for N is increased
Enablement	**N**: presents an action by R (including accepting an offer), unrealized with respect to the context of N **N + S**: R comprehending S increases R's potential ability to perform the action in N	R's potential ability to perform the action in N increases
Evidence	**N**: R might not believe N to a degree satisfactory to W **S**: R believes S or will find it credible **N + S**: R's comprehending S increases R's belief of N	R's belief of N is increased
Justify	**N + S**: R's comprehending S increases R's readiness to accept W's right to present N	R's readiness to accept W's right to present N is increased

Motivation	**N**: N is an action in which R is the actor (including accepting an offer), unrealized with respect to the context of N **N + S**: Comprehending S increases R's desire to perform action in N	R's desire to perform action in N is increased
Preparation	**N + S**: S precedes N in the text; S tends to make R more ready, interested or oriented for reading N	R is more ready, interested or oriented for reading N
Restatement	**N + S**: S restates N, where S and N are of comparable bulk; N is more central to W's purposes than S is	R recognizes S as a restatement of N
Summary	**N**: N must be more than one unit **N + S**: S presents a restatement of the content of N, that is shorter in bulk	R recognizes S as a shorter restatement of N

A.5.2. Subject-Matter Relations

Table A.2.: *Definitions of subject matter relations*

Relation	Constraints on N or S	Intention of W
Circumstance	**S**: S is not unrealized **N + S**: S sets a framework in the subject matter within which R is intended to interpret N	R recognizes that S provides the framework for interpreting N
Condition	**S**: S presents a hypothetical, future, or otherwise unrealized situation (relative to the situational context of S) **N + S**: Realization of N depends on realization of S	R recognizes how the realization of N depends on the realization of S
Elaboration	**N + S**: S presents additional detail about the situation or some element of subject matter which is presented in N or inferentially accessible in N in one or more of the ways listed below. In the list, if N presents the first member of any pair, then S includes the second: • set :: member • abstraction :: instance • whole :: part • process :: step • object :: attribute • generalization :: specific	R recognizes S as providing additional detail for N. R identifies the element of subject matter for which detail is provided.

Evaluation	**N + S**: S relates N to degree of W's positive regard toward N.	R recognizes that S assesses N and recognizes the value it assigns
Interpretation	**N + S**: S relates N to a framework of ideas not involved in N itself and not concerned with W's positive regard	R recognizes that S relates N to a framework of ideas not involved in the knowledge presented in N itself
Means	**N**: an activity **N + S**: S presents a method or instrument which tends to make realization of N more likely	R recognizes that the method or instrument in S tends to make realization of N more likely
Non-volitional Cause	**N**: N is not a volitional action **N + S**: S, by means other than motivating a volitional action, caused N; without the presentation of S, R might not know the particular cause of the situation; a presentation of N is more central than S to W's purposes in putting forth the N-S combination.	R recognizes S as a cause of N
Non-volitional Result	**S**: S is not a volitional action **N + S**: N caused S; presentation of N is more central to W's purposes in putting forth the N-S combination than is the presentation of S.	R recognizes that N could have caused the situation in S
Otherwise	**N**: N is an unrealized situation **S**: S is an unrealized situation **N + S**: realization of N prevents realization of S	R recognizes the dependency relation of prevention between the realization of N and the realization of S
Purpose	**N**: N is an activity; **S**: S is a situation that is unrealized **N + S**: S is to be realized through the activity in N	R recognizes that the activity in N is initiated in order to realize S
Solutionhood	**S**: S presents a problem **N + S**: N is a solution to the problem presented in S;	R recognizes N as a solution to the problem presented in S
Unconditional	**S**: S conceivably could affect the realization of N **N + S**: N does not depend on S	R recognizes that N does not depend on S
Unless	**N + S**: S affects the realization of N; N is realized provided that S is not realized	R recognizes that N is realized provided that S is not realized
Volitional Cause	**N**: N is a volitional action or else a situation that could have arisen from a volitional action **N + S**: S could have caused the agent of the volitional action in N to perform that action; without the presentation of S, R might not regard the action as motivated or know the particular motivation; N is more central to W's purposes in putting forth the N-S combination than S is.	R recognizes S as a cause for the volitional action in N

Volitional Result	**S**: S is a volitional action or a situation that could have arisen from a volitional action **N + S**: N could have caused S; presentation of N is more central to W's purposes than is presentation of S;	R recognizes that N could be a cause for the action or situation in S

A.5.3. Multinuclear Relations

Table A.3.: *Definitions of multinuclear relations*

Relation	Constraints on each N	Intention of W
Conjunction	The items are conjoined to form a unit in which each item plays a comparable role	R recognizes that the linked items are conjoined
Contrast	No more than two nuclei; the situations in these two nuclei are (a) comprehended as the same in many respects (b) comprehended as differing in a few respects and (c) compared with respect to one or more of these differences	R recognizes the comparability and the difference(s) yielded by the comparison is being made
Disjunction	An item presents a (not necessarily exclusive) alternative for the other(s)	R recognizes that the linked items are alternatives
Joint	none	none
List	An item comparable to others linked to it by the List relation	R recognizes the comparability of linked items
Multinuclear Restatement	An item is primarily a reexpression of one linked to it; the items are of comparable importance to the purposes of W	R recognizes the reexpression by the linked items
Sequence	There is a succession relationship between the situations in the nuclei	R recognizes the succession relationships among the nuclei.

A.6. “Gannet” Document Source Material

Original source material of the “Gannet” example as provided by Henschel et al. (2002).

A.6.1. Content

```
<content>
  <proposition id="latin" name="latin">
    <lf>
      <relation name="identifying">
        <domain>gannet</domain>
        <range>sula bassana</range>
      </relation>
    </lf>
    <realizations>
      <full>Gannet is in Latin Sula Bassana.</full>
      <ellipsis>Sula bassana</ellipsis>
      <table-form>
        <col-1-text>Gannet</col-1-text>
        <col-2-text>Sula bassana</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="family" name="family">
    <lf>
      <relation name="class-ascription">
        <domain>gannet</domain>
        <range>family sulidae</range>
      </relation>
    </lf>
    <realizations>
      <full>The Gannet belongs to the Sulidae.</full>
      <ellipsis>Family Sulidae</ellipsis>
      <table-form>
        <col-1-text>Gannet</col-1-text>
        <col-2-text>Family Sulidae</col-2-text>
      </table-form>
      </realizations>
  </proposition>
  <proposition id="s-24.5" name="presence">
    <lf/>
    <realizations>
      <full>The gannet is in Britain from January till December.</full>
      <ellipsis>In Britain  January  -  December</ellipsis>
      <table-form>
        <col-1-text>In Britain</col-1-text>
        <col-2-text>Jan - Dec</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="size" name="size">
```

```
    <lf/>
    <realizations>
      <full>The gannet is 87-100cm long.</full>
      <ellipsis>87-100cm</ellipsis>
      <table-form>
        <col-1-text>Size</col-1-text>
        <col-2-text>87-100cm</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="breeding-place" name="breeding-place">
    <lf/>
    <realizations>
      <full>Birds of the open ocean, Gannets breed on small islands off the NW coast of Europe.</full>
      <ellipsis>Breed on small islands off the NW coast of Europe</ellipsis>
      <table-form>
        <col-1-text>Breeding</col-1-text>
        <col-2-text>On small islands off the NW coast of Europe</col-2-text>
      </table-form>
      <graphics id="s-24.32" alt="map" src="images/holden-map.jpg"/>
    </realizations>
  </proposition>
  <proposition id="move" name="move">
    <lf/>
    <realizations>
      <full>They move away from land after nesting to winter at sea.</full>
      <ellipsis>Move away from land after nesting to winter at sea.</ellipsis>
      <table-form>
        <col-1-text>Habitat</col-1-text>
        <col-2-text>Move away from land after nesting to winter at sea.</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="migration" name="migration">
    <lf>
      <relation name="material">
        <actor>young gannets</actor>
        <relation name="circumstantial">
          <domain>young gannets</domain>
          <range>south as far as W Africa</range>
        </relation>
      </relation>
    </lf>
    <realizations>
      <full>The young migrate south as far as W Africa.</full>
      <ellipsis>Young migrate south as far as W Africa.</ellipsis>
      <table-form>
        <col-1-text>Migration</col-1-text>
        <col-2-text>Young migrate south as far as W Africa</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="food" name="food">
    <lf/>
    <realizations>
```

```
      <full>Gannets feed on fish by plunge-diving from 25m.</full>
      <ellipsis>Feed on fish by plunge-diving from 25m</ellipsis>
      <table-form>
        <col-1-text>Food</col-1-text>
        <col-2-text>Fish. Plunge-diving from 25m</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="colonies" name="colonies">
    <lf/>
    <realizations>
      <full>They nest  in large, noisy colonies.</full>
      <ellipsis>Nest in large noisy colonies</ellipsis>
      <table-form>
        <col-1-text>Habitat</col-1-text>
        <col-2-text>Large noisy colonies</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="nest" name="nest">
    <lf>
      <relation name="identifying">
        <domain>nest</domain>
        <range>pile of seaweed</range>
      </relation>
    </lf>
    <realizations>
      <full>The nest is a pile of seaweed.</full>
      <ellipsis>Nest pile of seaweed</ellipsis>
      <table-form>
        <col-1-text>Nest</col-1-text>
        <col-2-text>Pile of seaweed</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="egg" name="egg">
    <lf/>
    <realizations>
      <full> A single egg is incubated for 44 days.</full>
      <ellipsis>Single egg  incubated for 44 days</ellipsis>
      <table-form>
        <col-1-text>Egg</col-1-text>
        <col-2-text>A single egg, incubated for 44 days</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="parenting" name="parenting">
    <lf/>
    <realizations>
      <full> The young bird is fed by both parents and flies after 90 days. </full>
      <ellipsis>Young fed by both parents, flies after 90 days</ellipsis>
      <table-form>
        <col-1-text>Parenting</col-1-text>
        <col-2-text>Young fed by both parents, flies after 90 days</col-2-text>
      </table-form>
```

```
    </realizations>
  </proposition>
  <proposition id="size-by-comparison" name="size-by-comparison">
    <lf/>
    <realizations>
      <full>The gannet is larger than any gull.</full>
      <ellipsis>Larger than any gull</ellipsis>
      <table-form>
        <col-1-text>Size</col-1-text>
        <col-2-text>Larger than any gull</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="plumage-adult" name="plumage-adult">
    <lf/>
    <realizations>
      <full> The adult is white, has black wing-tips, and a yellow nape.</full>
      <ellipsis> Adult white, black wing-tips, yellow nape</ellipsis>
      <table-form>
        <col-1-text>Adult</col-1-text>
        <col-2-text> White, black wing-tips, yellow nape</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="plumage-juvenile" name="plumage-juvenile">
    <lf/>
    <realizations>
      <full>The juvenile gannet is grey, gradually becoming white over 5 years.</full>
      <ellipsis>Juvenile grey, gradually becoming white over 5 years</ellipsis>
      <table-form>
        <col-1-text>Juvenile</col-1-text>
        <col-2-text>Grey, gradually becoming white over 5 years</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="bil" name="bill">
    <lf/>
    <realizations>
      <full> The bill is dagger-like.</full>
      <ellipsis/>
      <table-form>
        <col-1-text>Bill</col-1-text>
        <col-2-text>Dagger-like</col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="flight-form" name="flight-form">
    <lf/>
    <realizations>
      <full>In flight, the gannet is cigar-shaped with long, narrow, black-tipped wings.</full>
      <ellipsis>In Flight, cigar-shaped with long, narrow, black-tipped wings.</ellipsis>
      <table-form>
        <col-1-text>In flight</col-1-text>
        <col-2-text>Cigar-shaped with long, narrow, black-tipped wings</col-2-text>
      </table-form>
```

```
    </realizations>
  </proposition>
  <proposition id="voice" name="voice">
    <lf/>
    <realizations>
      <full>The gannet is usually silent, growling urr when nesting.</full>
      <ellipsis>Usually silent, growling urr when nesting</ellipsis>
      <table-form>
        <col-1-text>Voice</col-1-text>
        <col-2-text>Usually silent, growling urr when nesting     </col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="lookalikes" name="lookalikes">
    <lf/>
    <realizations>
      <full> Lookalikes are Skuas, Gulls and Terns (pp 123-139).</full>
      <ellipsis>Lookalikes Skuas, Gulls and Terns (pp 123-139)  </ellipsis>
      <table-form>
        <col-1-text>Lookalikes</col-1-text>
        <col-2-text>Skuas, Gulls and Terns  </col-2-text>
      </table-form>
    </realizations>
  </proposition>
  <proposition id="bird" name="bird" mode="graphics">
    <lf/>
    <realizations>
      <full>Gannet</full>
      <graphics alt="photo" src="images/benson-image.gif">
        <caption>Adults at breeding colony</caption>
      </graphics>
    </realizations>
  </proposition>
  <proposition id="three-age-stages" name="three-age-stages">
    <lf/>
    <realizations>
      <graphics alt="three-age-stages" src="images/holden-drawing.jpg"/>
    </realizations>
  </proposition>
</content>
```

A.6.2. RST Structure

```
<rst-structure>
  <span id="holden-body" nucleus="holden-body-2" satellites="family" relation="elaboration"/>
  <span id="holden-body-2" nucleus="central" satellites="fact-file" relation="background">
    <title/>
  </span>
  <multi-span id="central" nuclei="lifestyle bird" relation="joint"/>
  <multi-span id="lifestyle" nuclei="habitat food fortpflanzung" relation="joint">
    <title>Lifestyle and habitat</title>
  </multi-span>
  <span id="habitat" nucleus="land" satellites="sea" relation="concession">
    <title>Habitat</title>
  </span>
  <span id="sea" nucleus="move" satellites="migration" relation="elaboration"/>
  <multi-span id="land" nuclei="breeding-place" relation="restatement"/>
  <multi-span id="fortpflanzung" nuclei="nest-1 egg parenting" relation="sequence"/>
  <span id="nest-1" nucleus="colonies" satellites="nest" relation="elaboration"/>
  <multi-span id="fact-file" nuclei="size-by-comparison plumage-adult plumage-juvenile bill flight-form
      voice lookalikes" relation="joint" topic="recognition-facts">
    <title>ID FACT FILE</title>
  </multi-span>
  <span id="span-24.1" nucleus="bird" satellites="s-24.32" relation="elaboration"/>
  <rst-root id="holden" nucleus="holden-body" satellites="latin" relation="background" topic="gannet">
    <title>Gannet</title>
  </rst-root>
</rst-structure>
```

B. Examples

This appendix compiles the example documents that are used in the thesis.

Specifically, Appendix B.1 provides the XML-based description of the “Yojijukugo” genre and its corresponding “two birds” document (Chap. 4); Appendix B.2 provides the XML-based description of the “Fieldguide” genre and its corresponding “Gannet” document (Chap. 5).

B.1. The “Yojijukugo” Genre Exemplified

B.1.1. The “two birds” Document

Basic Units

`Birds.basic_units.html`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
    <head>
        <title>Basic Units</title>
    </head>
    <body>
        <ul id="basic-units">
            <li id="u1" xlink:type="extended">
                <!-- one stone -->
                <span xlink:type="arc" xlink:show="replace"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
                <span xlink:type="locator" xlink:label="_" xlink:href="Birds.content.ja.html#ja1"
                      xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"></span>
                <span xlink:type="locator" xlink:label="_" xlink:href="Birds.content.en.html#en1"
                      xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"></span>
            </li>
            <li id="u2" xlink:type="extended">
                <!-- two birds -->
                <span xlink:type="arc" xlink:show="replace"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
                <span xlink:type="locator" xlink:label="_" xlink:href="Birds.content.ja.html#ja2"
                      xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"></span>
                <span xlink:type="locator" xlink:label="_" xlink:href="Birds.content.en.html#en2"
                      xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"></span>
            </li>
        </ul>
    </body>
</html>
```

Japanese Content Nodes

`Birds.content.ja.html`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Content Base &mdash; Japanese</title>
    </head>
    <body>
        <ul>
            <li id="ja1">&#19968;&#30707;</li><!-- isseki -->
            <li id="ja2">&#20108;&#40165;</li><!-- nichou -->
        </ul>
    </body>
</html>
```

English Content Nodes

`Birds.content.en.html`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Content Base &mdash; English</title>
    </head>
    <body>
        <ul>
            <li id="en1">one stone</li>
            <li id="en2">two birds</li>
        </ul>
    </body>
</html>
```

Japanese Output Structure

`Birds.ja.struct.html`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <title>Japanese Structure</title>
  </head>
  <body>
    <ul id="struct">
      <li xlink:type="extended">
        <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
        <span xlink:type="resource" xlink:label="parent" id="node_1"></span>
        <span xlink:type="locator"  xlink:label="child" id="content_1.1"
              xlink:href="Birds.basic_units.html#u1"
              xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"></span>
        <span xlink:type="locator"  xlink:label="child" id="content_1.2"
              xlink:href="Birds.basic_units.html#u2"
              xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"></span>
      </li>
    </ul>
  </body>
</html>
```

Japanese Output Labels

`Birds.ja.labels.html`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <title>Japanese Labels</title>
  </head>
  <body>
    <ol xml:base="Birds.ja.struct.html" id="labels">
      <li xlink:type="extended">
        <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="resource" xlink:label="type"
              xlink:role="http://www.w3.org/2001/XMLSchema#QName">html:p</span>
      </li>
      <li xlink:type="extended">
        <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
        <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
```

```
17          <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
18          <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">xmlns:html</span>
19          <span xlink:type="resource" xlink:label="value"
                  xlink:role="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/1999/xhtml</span>
20        </li>
21        <li xlink:type="extended">
22          <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
23          <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
24          <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
25          <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">html:lang</span>
26          <span xlink:type="resource" xlink:label="value"
                  xlink:role="http://www.w3.org/2001/XMLSchema#string">ja</span>
27        </li>
28      </ol>
29    </body>
30  </html>
```

English Output Structure

`Birds.en.struct.html`

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
4     <head>
5       <title>English Structure</title>
6     </head>
7     <body>
8       <ul id="struct">
9         <li xlink:type="extended">
10          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
11          <span xlink:type="resource" xlink:label="parent" id="node_1"></span>
12          <span xlink:type="locator"  xlink:label="child" id="content_1.1"
                  xlink:href="Birds.basic_units.html#u2"
                  xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"></span>
13          <span xlink:type="locator"  xlink:label="child" id="content_1.2"
                  xlink:href="Birds.basic_units.html#u1"
                  xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"></span>
14        </li>
15      </ul>
16    </body>
17  </html>
```

English Output Labels

`Birds.en.labels.html`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <title>English Labels</title>
  </head>
  <body>
    <ol xml:base="Birds.en.struct.html" id="labels">
      <li xlink:type="extended">
        <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="resource" xlink:label="type"
              xlink:role="http://www.w3.org/2001/XMLSchema#QName">html:p</span>
      </li>
      <li xlink:type="extended">
        <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
        <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
        <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1"
              xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
        <span xlink:type="resource" xlink:label="type"
              xlink:role="http://www.w3.org/2001/XMLSchema#QName">xmlns:html</span>
        <span xlink:type="resource" xlink:label="value"
              xlink:role="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/1999/xhtml</span>
      </li>
      <li xlink:type="extended">
        <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
        <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
        <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1"
              xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
        <span xlink:type="resource" xlink:label="type"
              xlink:role="http://www.w3.org/2001/XMLSchema#QName">html:lang</span>
        <span xlink:type="resource" xlink:label="value"
              xlink:role="http://www.w3.org/2001/XMLSchema#string">en</span>
      </li>
    </ol>
  </body>
</html>
```

Multistructured Document Description

`Birds.doc.out.html`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
    <head>
        <title>Document</title>
    </head>
    <body>
        <div xlink:type="extended">
            <!-- Structure: Japanese output -->
            <span xlink:type="arc" xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
                  xlink:show="embed" xlink:actuate="onLoad" xlink:from="struct.ja"
                  xlink:to="labels.ja"></span>
            <span xlink:type="locator" xlink:label="struct.ja" xlink:title="Japanese Structure"
                  xlink:href="Birds.ja.struct.html#node_1"></span>
            <span xlink:type="locator" xlink:label="labels.ja" xlink:title="Japanese Labels"
                  xlink:role="http://www.w3.org/1999/xhtml"
                  xlink:href="Birds.ja.labels.html#labels"></span>
            <!-- Structure: English output -->
            <span xlink:type="arc" xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
                  xlink:show="embed" xlink:actuate="onLoad" xlink:from="struct.en"
                  xlink:to="labels.en"></span>
            <span xlink:type="locator" xlink:label="struct.en" xlink:title="English Structure"
                  xlink:href="Birds.en.struct.html#node_1"></span>
            <span xlink:type="locator" xlink:label="labels.en" xlink:title="English Labels"
                  xlink:role="http://www.w3.org/1999/xhtml"
                  xlink:href="Birds.en.labels.html#labels"></span>
            <!-- Content Base -->
            <span xlink:type="arc" xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
                  xlink:show="none" xlink:actuate="none" xlink:from="content" xlink:to="units"></span>
            <span xlink:type="locator" xlink:label="units" xlink:title="Basic Units"
                  xlink:href="Birds.basic_units.html#basic-units"></span>
            <span xlink:type="locator" xlink:label="content" xlink:title="English"
                  xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"
                  xlink:href="Birds.content.en.html"></span>
            <span xlink:type="locator" xlink:label="content" xlink:title="Japanese"
                  xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"
                  xlink:href="Birds.content.ja.html"></span>
        </div>
    </body>
</html>
```

B.1.2. The “Yojijukugo” Genre

Content Structure

`Birds.RST.struct.html`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
    <head>
        <title>RST Structure</title>
    </head>
    <body>
        <ul id="struct">
            <li xlink:type="extended">
                <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                      xlink:actuate="onLoad"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
                <span xlink:type="resource" xlink:label="parent" id="node_1"></span>
                <span xlink:type="locator" xlink:label="child" id="edge_1.1" xlink:href="#node_1.1"></span>
                <span xlink:type="locator" xlink:label="child" id="edge_1.2" xlink:href="#node_1.2"></span>
            </li>
            <li xlink:type="extended">
                <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                      xlink:actuate="onLoad"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
                <span xlink:type="resource" xlink:label="parent" id="node_1.1"></span>
                <span xlink:type="locator" xlink:label="child" id="edge_1.1.1"
                      xlink:href="Birds.basic_units.html#u1"></span>
            </li>
            <li xlink:type="extended">
                <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                      xlink:actuate="onLoad"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
                <span xlink:type="resource" xlink:label="parent" id="node_1.2"></span>
                <span xlink:type="locator" xlink:label="child" id="edge_1.2.1"
                      xlink:href="Birds.basic_units.html#u2"></span>
            </li>
        </ul>
    </body>
</html>
```

Content Labels

`Birds.RST.labels.html`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
    <head>
        <title>RST Labels</title>
    </head>
    <body>
        <ol xml:base="Birds.RST.struct.html" id="labels">
            <li xlink:type="extended">
                <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
                <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
                <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1"
                      xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
                <span xlink:type="resource" xlink:label="type"
                      xlink:role="http://www.w3.org/2001/XMLSchema#QName">xmlns:rst</span>
                <span xlink:type="resource" xlink:label="value"
                      xlink:role="http://www.w3.org/2001/XMLSchema#string">http://www.sfu.ca/rst</span>
            </li>
            <li xlink:type="extended">
                <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
                      xlink:actuate="onLoad"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
                <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1"
                      xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
                <span xlink:type="resource" xlink:label="type"
                      xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:group</span>
            </li>
            <li xlink:type="extended">
                <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
                      xlink:actuate="onLoad"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
                <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1.1"
                      xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
                <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1.2"
                      xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
                <span xlink:type="resource" xlink:label="type"
                      xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:segment</span>
            </li>
            <li xlink:type="extended">
                <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                      xlink:actuate="onLoad"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
                <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                      xlink:actuate="onLoad"
                      xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
                <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1.1"
                      xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
                <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1.2"
                      xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
```

```
32                <span xlink:type="resource" xlink:label="type"
                        xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:relname</span>
33                <span xlink:type="resource" xlink:label="value"
                        xlink:role="http://www.w3.org/2001/XMLSchema#string">joint</span>
34            </li>
35            <li xlink:type="extended">
36                <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                        xlink:actuate="onLoad"
                        xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
37                <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                        xlink:actuate="onLoad"
                        xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
38                <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1.1"
                        xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
39                <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1.2"
                        xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
40                <span xlink:type="resource" xlink:label="type"
                        xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:reltype</span>
41                <span xlink:type="resource" xlink:label="value"
                        xlink:role="http://www.w3.org/2001/XMLSchema#string">multinuc</span>
42            </li>
43        </ol>
44    </body>
45 </html>
```

Multistructured Document Description

`Birds.doc.html`

```
1  <?xml version="1.0"?>
2  <!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
4      <head>
5          <title>Document</title>
6      </head>
7      <body>
8          <div xlink:type="extended">
9              <!-- Structure: RST -->
10             <span xlink:type="arc" xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
                       xlink:show="embed" xlink:actuate="onLoad" xlink:from="struct.RST"
                       xlink:to="labels.RST"></span>
11             <span xlink:type="locator" xlink:label="struct.RST" xlink:title="RST Structure"
                       xlink:href="Birds.RST.struct.html#node_1"></span>
12             <span xlink:type="locator" xlink:label="labels.RST" xlink:title="RST Labels"
                       xlink:role="http://www.sfu.ca/rst" xlink:href="Birds.RST.labels.html#labels"></span>
13             <!-- Content Base -->
14             <span xlink:type="arc" xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
                       xlink:show="none" xlink:actuate="none" xlink:from="content" xlink:to="units"></span>
15             <span xlink:type="locator" xlink:label="units" xlink:title="Basic Units"
                       xlink:href="Birds.basic_units.html#basic-units"></span>
16             <span xlink:type="locator" xlink:label="content" xlink:title="English"
                       xlink:role="http://www.iana.org/assignments/language-subtag-registry#en"
```

```
                xlink:href="Birds.content.en.html"></span>
17          <span xlink:type="locator" xlink:label="content" xlink:title="Japanese"
                xlink:role="http://www.iana.org/assignments/language-subtag-registry#ja"
                xlink:href="Birds.content.ja.html"></span>
18      </div>
19  </body>
20 </html>
```

Content Schema

Yoji.content.sch

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <schema queryBinding="xslt2"
3      xmlns="http://purl.oclc.org/dsdl/schematron"
4      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
5
6      <xsl:include href="transform/rst/rst-functions.xsl"/>
7      <include href="transform/rst/rst.sch#gen"/>
8
9      <ns uri="http://dmt.uni-wuppertal.de/schlupko/xlink4dags" prefix="dag"/>
10     <ns uri="http://www.sfu.ca/rst" prefix="rst"/>
11
12     <let name="genre" value="'Yojijukugo'"/>
13
14     <pattern>
15         <title>Yojijukugo Genre - Content Specifics</title>
16         <rule context="/rst:group">
17             <assert test="count(*) = 2">
18                 A "<value-of select="$genre"/>"'s top-level group has exactly two children.</assert>
19             <report test="*[not( self::rst:segment )]">
20                 A "<value-of select="$genre"/>"'s top-level group is composed of content segments,
                       only.</report>
21             <report test="*[not( rst:is_multinuclear( ., 'joint' ) )]">
22                 A "<value-of select="$genre"/>"'s top-level group is composed of nuclei that are in
                       "joint" relation to each other, only.</report>
23         </rule>
24     </pattern>
25
26 </schema>
```

Japanese Subgenre Pipeline

`Yoji.content2form.ja.xpl`

```
<?xml version="1.0" encoding="UTF-8"?>
<p:declare-step version="1.0"
    xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:c="http://www.w3.org/ns/xproc-step"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags">

    <p:option name="input-file"  select="'Birds.doc.html'"/>
    <p:option name="output-file" select="'Birds.doc.ja.html'"/>

    <p:import href="transform/stdoff/xlink-validate.xpl"/>

    <p:load name="dag-input">
        <p:with-option name="href" select="$input-file"/>
    </p:load>

    <dag:validate-with-schematron>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="schema">
            <p:document href="Yoji.content.sch"/>
        </p:input>
    </dag:validate-with-schematron>
    <p:sink/>

    <p:xslt>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="stylesheet">
            <p:document href="Yoji.content2form.ja.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
    </p:xslt>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="transform/stdoff/xml-cleanup.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
    </p:xslt>

    <p:store>
        <p:with-option name="href" select="$output-file"/>
    </p:store>

</p:declare-step>
```

Japanese Subgenre Transformation

`Yoji.content2form.ja.xsl`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:rst="http://www.sfu.ca/rst"
    xmlns:html="http://www.w3.org/1999/xhtml"
    xmlns:yoji="https://en.wikipedia.org/wiki/Yojijukugo"
    exclude-result-prefixes="#all">

    <xsl:import href="transform/stdoff/stdoff-transform.xsl"/>
    <xsl:import href="transform/rst/rst-functions.xsl"/>

    <xsl:output doctype-public="-//W3C//DTD XHTML 1.1//EN"
          doctype-system="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" indent="yes"/>

    <!-- HTML main structure -->

    <xsl:template match="/rst:group" priority="10">
        <xsl:element name="html:html">
            <xsl:element name="html:head">
                <xsl:element name="html:title">
                    <xsl:call-template name="keep-endtag"/>
                </xsl:element>
                <xsl:element name="html:meta">
                    <xsl:attribute name="http-equiv" select="'content-type'"/>
                    <xsl:attribute name="content" select="'text/html; charset=utf-8'"/>
                </xsl:element>
            </xsl:element>
            <xsl:element name="html:body">
                <xsl:next-match/>
            </xsl:element>
        </xsl:element>
    </xsl:template>

    <xsl:template name="keep-endtag">
        <xsl:text>&#x200B;</xsl:text>
    </xsl:template>

    <!-- content -->

    <xsl:template match="/rst:group">
        <xsl:element name="html:p">
            <xsl:apply-templates>
                <xsl:with-param name="content-mode"
                      select="'http://www.iana.org/assignments/language-subtag-registry#ja'" tunnel="yes"/>
            </xsl:apply-templates>
        </xsl:element>
    </xsl:template>

</xsl:stylesheet>
```

English Subgenre Pipeline

`Yoji.content2form.en.xpl`

```
<?xml version="1.0" encoding="UTF-8"?>
<p:declare-step version="1.0"
    xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:c="http://www.w3.org/ns/xproc-step"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags">

    <p:option name="input-file"  select="'Birds.doc.html'"/>
    <p:option name="output-file" select="'Birds.doc.en.html'"/>

    <p:import href="transform/stdoff/xlink-validate.xpl"/>

    <p:load name="dag-input">
        <p:with-option name="href" select="$input-file"/>
    </p:load>

    <dag:validate-with-schematron>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="schema">
            <p:document href="Yoji.content.sch"/>
        </p:input>
    </dag:validate-with-schematron>
    <p:sink/>

    <p:xslt>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="stylesheet">
            <p:document href="Yoji.content2form.en.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
    </p:xslt>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="transform/stdoff/xml-cleanup.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
    </p:xslt>

    <p:store>
        <p:with-option name="href" select="$output-file"/>
    </p:store>

</p:declare-step>
```

English Subgenre Transformation

`Yoji.content2form.en.xsl`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:rst="http://www.sfu.ca/rst"
    xmlns:html="http://www.w3.org/1999/xhtml"
    xmlns:yoji="https://en.wikipedia.org/wiki/Yojijukugo"
    exclude-result-prefixes="#all">

    <xsl:import href="transform/stdoff/stdoff-transform.xsl"/>
    <xsl:import href="transform/rst/rst-functions.xsl"/>

    <xsl:output doctype-public="-//W3C//DTD XHTML 1.1//EN"
          doctype-system="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" indent="yes"/>

    <!-- HTML main structure -->

    <xsl:template match="/rst:group" priority="10">
        <xsl:element name="html:html">
            <xsl:element name="html:head">
                <xsl:element name="html:title">
                    <xsl:call-template name="keep-endtag"/>
                </xsl:element>
                <xsl:element name="html:meta">
                    <xsl:attribute name="http-equiv" select="'content-type'"/>
                    <xsl:attribute name="content" select="'text/html; charset=utf-8'"/>
                </xsl:element>
            </xsl:element>
            <xsl:element name="html:body">
                <xsl:next-match/>
            </xsl:element>
        </xsl:element>
    </xsl:template>

    <xsl:template name="keep-endtag">
        <xsl:text>&#x200B;</xsl:text>
    </xsl:template>

    <!-- content -->

    <xsl:template match="/rst:group">
        <xsl:element name="html:p">
            <xsl:apply-templates select="*[2], yoji:separator(), *[1]">
                <xsl:with-param name="content-mode"
                      select="'http://www.iana.org/assignments/language-subtag-registry#en'" tunnel="yes"/>
            </xsl:apply-templates>
        </xsl:element>
    </xsl:template>

    <xsl:function name="yoji:separator">
        <xsl:element name="html:span">
            <xsl:text>, </xsl:text>
```

```
50          </xsl:element>
51      </xsl:function>
52
53  </xsl:stylesheet>
```

B.2. The “Fieldguide” Genre Exemplified

B.2.1. The “Gannet” Document

Content Base

Basic Units `Gannet.basic_units.xhtml`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <title>Basic Units</title>
  </head>
  <body>
    <ul id="basic-units">
      <li id="u1" xlink:type="extended">
        <!-- Gannet -->
        <span xlink:type="arc" xlink:show="replace"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t1"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l1"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t1"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.img.xhtml#i1"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a1"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.3d.xhtml#m1"
              xlink:role="http://www.web3d.org/specifications/x3d-namespace"></span>
      </li>
      <li id="u2" xlink:type="extended">
        <!-- Sula bassana -->
        <span xlink:type="arc" xlink:show="replace"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t2"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l2"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t2"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a2"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
      </li>
      <li id="u3" xlink:type="extended">
        <!-- Family Sulidae -->
        <span xlink:type="arc" xlink:show="replace"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t3"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l3"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
```

```
32        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t3"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
33        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a3"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
34      </li>
35      <li id="u4" xlink:type="extended">
36        <!-- In Britain January-December -->
37        <span xlink:type="arc" xlink:show="replace"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
38        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t4"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
39        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l4"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
40        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t4"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
41        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a4"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
42      </li>
43      <li id="u5" xlink:type="extended">
44        <!-- 87-100cm -->
45        <span xlink:type="arc" xlink:show="replace"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
46        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t5"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
47        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l5"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
48        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t5"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
49        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a5"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
50      </li>
51      <li id="u6" xlink:type="extended">
52        <!-- Breed on small islands off the NW coast of Europe -->
53        <span xlink:type="arc" xlink:show="replace"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
54        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t6"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
55        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l6"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
56        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t6"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
57        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.img.xhtml#i6"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod"></span>
58        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a6"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
59        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.video.xhtml#v6"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod"></span>
60      </li>
61      <li id="u7" xlink:type="extended">
62        <!-- Move away from land after nesting to winter at sea -->
63        <span xlink:type="arc" xlink:show="replace"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
64        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t7"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
```

```
65        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l7"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
66        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t7"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
67        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a7"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
68        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.video.xhtml#v7"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod"></span>
69      </li>
70      <li id="u8" xlink:type="extended">
71        <!-- Young migrate south as far as W Africa -->
72        <span xlink:type="arc" xlink:show="replace"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
73        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t8"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
74        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l8"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
75        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t8"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
76        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a8"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
77        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.video.xhtml#v8"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod"></span>
78      </li>
79      <li id="u9" xlink:type="extended">
80        <!-- Feed on fish by plunge-diving from 25m -->
81        <span xlink:type="arc" xlink:show="replace"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
82        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t9"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
83        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l9"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
84        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t9"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
85        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a9"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
86        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.video.xhtml#v9"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod"></span>
87      </li>
88      <li id="u10" xlink:type="extended">
89        <!-- Nest in large noisy colonies -->
90        <span xlink:type="arc" xlink:show="replace"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
91        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t10"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
92        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l10"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
93        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t10"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
94        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a10"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
95        <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.video.xhtml#v10"
              xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod"></span>
96      </li>
97      <li id="u11" xlink:type="extended">
```

```
98          <!-- Nest pile of seaweed -->
99          <span xlink:type="arc" xlink:show="replace"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
100         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t11"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
101         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l11"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
102         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t11"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
103         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a11"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
104         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.video.xhtml#v11"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod"></span>
105       </li>
106       <li id="u12" xlink:type="extended">
107         <!-- Single egg incubated for 44 days -->
108         <span xlink:type="arc" xlink:show="replace"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
109         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t12"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
110         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l12"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
111         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t12"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
112         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a12"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
113         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.video.xhtml#v12"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod"></span>
114       </li>
115       <li id="u13" xlink:type="extended">
116         <!-- Young fed by both parents, flies after 90 days -->
117         <span xlink:type="arc" xlink:show="replace"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
118         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t13"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
119         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l13"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
120         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t13"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
121         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a13"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
122         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.video.xhtml#v13"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod"></span>
123       </li>
124       <li id="u14" xlink:type="extended">
125         <!-- Larger than any gull -->
126         <span xlink:type="arc" xlink:show="replace"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
127         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t14"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
128         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l14"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
129         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t14"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
```

```
130       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a14"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
131     </li>
132     <li id="u15" xlink:type="extended">
133       <!-- Adult white, black wing-tips, yellow nape -->
134       <span xlink:type="arc" xlink:show="replace"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
135       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t15"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
136       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l15"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
137       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t15"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
138       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a15"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
139     </li>
140     <li id="u16" xlink:type="extended">
141       <!-- Juvenile grey, gradually becoming white over 5 years -->
142       <span xlink:type="arc" xlink:show="replace"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
143       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t16"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
144       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l16"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
145       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t16"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
146       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.img.xhtml#i16"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod"></span>
147       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a16"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
148     </li>
149     <li id="u17" xlink:type="extended">
150       <!-- Bill dagger-like -->
151       <span xlink:type="arc" xlink:show="replace"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
152       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t17"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
153       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l17"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
154       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t17"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
155       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a17"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
156     </li>
157     <li id="u18" xlink:type="extended">
158       <!-- In Flight, cigar-shaped with long, narrow, black-tipped wings -->
159       <span xlink:type="arc" xlink:show="replace"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
160       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t18"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
161       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l18"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
162       <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t18"
                xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
```

```
163         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a18"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
164       </li>
165       <li id="u19" xlink:type="extended">
166         <!-- Usually silent, growling "urr" when nesting -->
167         <span xlink:type="arc" xlink:show="replace"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
168         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t19"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
169         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l19"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
170         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t19"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
171         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a19"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
172       </li>
173       <li id="u20" xlink:type="extended">
174         <!-- Lookalikes Skuas, Gulls and Terns -->
175         <span xlink:type="arc" xlink:show="replace"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt"></span>
176         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.text.xhtml#t20"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"></span>
177         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.list.xhtml#l20"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"></span>
178         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.table.xhtml#t20"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"></span>
179         <span xlink:type="locator" xlink:label="_" xlink:href="Gannet.content.audio.xhtml#a20"
                  xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"></span>
180       </li>
181     </ul>
182   </body>
183 </html>
```

Text Realizations `Gannet.content.text.xhtml`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Content Base &mdash; Text</title>
  </head>
  <body>
    <ul>
      <li id="t1">The Gannet </li>
      <li id="t2">Gannet is in Latin Sula Bassana. </li>
      <li id="t3">The Gannet belongs to the Sulidae. </li>
      <li id="t4">The gannet is in Britain from January till December. </li>
      <li id="t5">The gannet is 87&ndash;100<abbr title="centimeters">cm</abbr> long. </li>
      <li id="t6">Birds of the open ocean, Gannets breed on small islands off the <abbr
          title="North-West">NW</abbr> coast of Europe. </li>
      <li id="t7">They move away from land after nesting to winter at sea. </li>
      <li id="t8">The young migrate south as far as <abbr title="West">W</abbr> Africa. </li>
      <li id="t9">Gannets feed on fish by plunge-diving from 25<abbr title="meters">m</abbr>. </li>
      <li id="t10">They nest in large, noisy colonies. </li>
      <li id="t11">The nest is a pile of seaweed. </li>
      <li id="t12">A single egg is incubated for 44 days. </li>
      <li id="t13">The young bird is fed by both parents and flies after 90 days. </li>
      <li id="t14">The gannet is larger than any gull. </li>
      <li id="t15">The adult is white, has black wing-tips, and a yellow nape. </li>
      <li id="t16">The juvenile gannet is grey, gradually becoming white over 5 years. </li>
      <li id="t17">The bill is dagger-like. </li>
      <li id="t18">In flight, the gannet is cigar-shaped with long, narrow, black-tipped wings. </li>
      <li id="t19">The gannet is usually silent, growling <q>urr</q> when nesting. </li>
      <li id="t20">Lookalikes are Skuas, Gulls and Terns. </li>
    </ul>
  </body>
</html>
```

List Realizations `Gannet.content.list.xhtml`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Content Base &mdash; List</title>
  </head>
  <body>
    <ul>
      <li id="l1">Gannet</li>
      <li id="l2">Sula bassana</li>
      <li id="l3">Family Sulidae</li>
      <li id="l4">In Britain January&ndash;December</li>
      <li id="l5">87&ndash;100<abbr title="centimeters">cm</abbr></li>
      <li id="l6">Breed on small islands off the <abbr title="North-West">NW</abbr> coast of Europe</li>
      <li id="l7">Move away from land after nesting to winter at sea</li>
      <li id="l8">Young migrate south as far as <abbr title="West">W</abbr> Africa</li>
      <li id="l9">Feed on fish by plunge-diving from 25<abbr title="meters">m</abbr></li>
      <li id="l10">Nest in large noisy colonies</li>
      <li id="l11">Nest pile of seaweed</li>
      <li id="l12">Single egg incubated for 44 days</li>
      <li id="l13">Young fed by both parents, flies after 90 days</li>
      <li id="l14">Larger than any gull</li>
      <li id="l15">Adult white, black wing-tips, yellow nape</li>
      <li id="l16">Juvenile grey, gradually becoming white over 5 years</li>
      <li id="l17">Bill dagger-like</li>
      <li id="l18">In Flight, cigar-shaped with long, narrow, black-tipped wings</li>
      <li id="l19">Usually silent, growling <q>urr</q> when nesting</li>
      <li id="l20">Lookalikes Skuas, Gulls and Terns</li>
    </ul>
  </body>
</html>
```

Table Realizations `Gannet.content.table.xhtml`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Content Base &mdash; Table</title>
  </head>
  <body>
    <table>
      <tr id="t1">
        <th>Name</th>
        <td>Gannet</td>
      </tr>
      <tr id="t2">
        <th>Latin</th>
        <td>Sula bassana</td>
      </tr>
      <tr id="t3">
        <th>Family</th>
        <td>Sulidae</td>
      </tr>
      <tr id="t4">
        <th>In Britain</th>
        <td>Jan&ndash;Dec</td>
      </tr>
      <tr id="t5">
        <th>Size</th>
        <td>87&ndash;100<abbr title="centimeters">cm</abbr></td>
      </tr>
      <tr id="t6">
        <th>Breeding</th>
        <td>On small islands off the <abbr title="North-West">NW</abbr> coast of Europe</td>
      </tr>
      <tr id="t7">
        <th>Habitat</th>
        <td>Move away from land after nesting to winter at sea</td>
      </tr>
      <tr id="t8">
        <th>Migration</th>
        <td>Young migrate south as far as <abbr title="West">W</abbr> Africa</td>
      </tr>
      <tr id="t9">
        <th>Food</th>
        <td>Fish. Plunge-diving from 25<abbr title="meters">m</abbr></td>
      </tr>
      <tr id="t10">
        <th>Habitat</th>
        <td>Large noisy colonies</td>
      </tr>
      <tr id="t11">
```

```
        <th>Nest</th>
        <td>Pile of seaweed</td>
      </tr>
      <tr id="t12">
        <th>Egg</th>
        <td>A single egg, incubated for 44 days</td>
      </tr>
      <tr id="t13">
        <th>Parenting</th>
        <td>Young fed by both parents, flies after 90 days</td>
      </tr>
      <tr id="t14">
        <th>Size</th>
        <td>Larger than any gull</td>
      </tr>
      <tr id="t15">
        <th>Adult</th>
        <td>White, black wing-tips, yellow nape</td>
      </tr>
      <tr id="t16">
        <th>Juvenile</th>
        <td>Grey, gradually becoming white over 5 years</td>
      </tr>
      <tr id="t17">
        <th>Bill</th>
        <td>Dagger-like</td>
      </tr>
      <tr id="t18">
        <th>In flight</th>
        <td>Cigar-shaped with long, narrow, black-tipped wings</td>
      </tr>
      <tr id="t19">
        <th>Voice</th>
        <td>Usually silent, growling <q>urr</q> when nesting</td>
      </tr>
      <tr id="t20">
        <th>Lookalikes</th>
        <td>Skuas, Gulls and Terns</td>
      </tr>
    </table>
  </body>
</html>
```

Image Realizations `Gannet.content.img.xhtml`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Content Base &mdash; Image</title>
  </head>
  <body>
    <ul>
      <li id="i1"><img alt="Adults at breeding colony" src="media/images/gannet.jpg"/></li>
      <li id="i6"><img alt="map" src="media/images/holden-map.jpg"/></li>
      <li id="i16"><img alt="three-age-stages" src="media/images/holden-drawing.jpg"/></li>
    </ul>
  </body>
</html>
```

Audio Realizations `Gannet.content.audio.xhtml`

```
<?xml version="1.0"?>
<!DOCTYPE html [<!ATTLIST li id ID #IMPLIED>]>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Content Base &#8212; Audio</title>
  </head>
  <body>
    <ul>
      <li id="a1"><audio controls="controls"><source src="media/audio/1.wav"
          type="audio/wav"/></audio></li>
      <li id="a2"><audio controls="controls"><source src="media/audio/2.wav"
          type="audio/wav"/></audio></li>
      <li id="a3"><audio controls="controls"><source src="media/audio/3.wav"
          type="audio/wav"/></audio></li>
      <li id="a4"><audio controls="controls"><source src="media/audio/4.wav"
          type="audio/wav"/></audio></li>
      <li id="a5"><audio controls="controls"><source src="media/audio/5.wav"
          type="audio/wav"/></audio></li>
      <li id="a6"><audio controls="controls"><source src="media/audio/6.wav"
          type="audio/wav"/></audio></li>
      <li id="a7"><audio controls="controls"><source src="media/audio/7.wav"
          type="audio/wav"/></audio></li>
      <li id="a8"><audio controls="controls"><source src="media/audio/8.wav"
          type="audio/wav"/></audio></li>
      <li id="a9"><audio controls="controls"><source src="media/audio/9.wav"
          type="audio/wav"/></audio></li>
      <li id="a10"><audio controls="controls"><source src="media/audio/10.wav"
          type="audio/wav"/></audio></li>
      <li id="a11"><audio controls="controls"><source src="media/audio/11.wav"
          type="audio/wav"/></audio></li>
      <li id="a12"><audio controls="controls"><source src="media/audio/12.wav"
          type="audio/wav"/></audio></li>
      <li id="a13"><audio controls="controls"><source src="media/audio/13.wav"
          type="audio/wav"/></audio></li>
      <li id="a14"><audio controls="controls"><source src="media/audio/14.wav"
          type="audio/wav"/></audio></li>
      <li id="a15"><audio controls="controls"><source src="media/audio/15.wav"
          type="audio/wav"/></audio></li>
      <li id="a16"><audio controls="controls"><source src="media/audio/16.wav"
          type="audio/wav"/></audio></li>
      <li id="a17"><audio controls="controls"><source src="media/audio/17.wav"
          type="audio/wav"/></audio></li>
      <li id="a18"><audio controls="controls"><source src="media/audio/18.wav"
          type="audio/wav"/></audio></li>
      <li id="a19"><audio controls="controls"><source src="media/audio/19.wav"
          type="audio/wav"/></audio></li>
      <li id="a20"><audio controls="controls"><source src="media/audio/20.wav"
          type="audio/wav"/></audio></li>
    </ul>
```

```
  </body>
</html>
```

Video Realizations `Gannet.content.video.xhtml`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html [<!ATTLIST li id ID #IMPLIED>]>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Content Base &#8212; Video</title>
  </head>
  <body>
    <ul>
      <li id="v6"><video width="480" height="246" controls="controls"><source src="media/video/6.mp4"
          type="video/mp4"/></video></li>
      <li id="v7"><video width="480" height="246" controls="controls"><source src="media/video/7.mp4"
          type="video/mp4"/></video></li>
      <li id="v8"><video width="480" height="246" controls="controls"><source src="media/video/8.mp4"
          type="video/mp4"/></video></li>
      <li id="v9"><video width="480" height="246" controls="controls"><source src="media/video/9.mp4"
          type="video/mp4"/></video></li>
      <li id="v10"><video width="480" height="246" controls="controls"><source src="media/video/10.mp4"
          type="video/mp4"/></video></li>
      <li id="v11"><video width="480" height="246" controls="controls"><source src="media/video/11.mp4"
          type="video/mp4"/></video></li>
      <li id="v12"><video width="480" height="246" controls="controls"><source src="media/video/12.mp4"
          type="video/mp4"/></video></li>
      <li id="v13"><video width="480" height="246" controls="controls"><source src="media/video/13.mp4"
          type="video/mp4"/></video></li>
    </ul>
  </body>
</html>
```

Spatial Realizations `Gannet.content.3d.xhtml`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html [<!ATTLIST Group DEF ID #IMPLIED>]>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Content Base &#8212; 3D</title>
    <script type="text/javascript" src="script/x3dom-1.7.0/x3dom.js"></script>
    <script type="text/javascript" src="script/x3dom-1.7.0/components/CADGeometry.js"></script>
    <link rel="stylesheet" type="text/css" href="script/x3dom-1.7.0/x3dom.css"></link>
  </head>
  <body>
    <X3D xmlns="http://www.web3d.org/specifications/x3d-namespace" width="800px" height="500px">
      <Scene>
        <Group DEF="m1">
          <Shape>
            <Appearance>
              <Material ambientIntensity='0.333' diffuseColor='1 1 1' shininess='0.098'
                        specularColor='0.401 0.401 0.401'/>
            </Appearance>
            <IndexedTriangleSet solid="false" colorPerVertex='false' index='0 1 2 2 1 3 4 5 6 7 6 5 6 7 8
                      9 8 7 10 11 12 12 11 13 10 14 11 11 15 13 14 16 11 17 18 19 19 18 20 20 18 21 22 21
                      18 23 18 24 17 24 18 22 25 21 26 22 18 27 ...
              '>
              <Coordinate point='-3.15473 6.201 -0.426486 -3.12203 6.18898 -0.439058 -3.15643 6.22113
                        -0.41633 -3.08414 6.27276 -0.41633 -3.08414 6.27276 -0.41633 -3.05157 6.34475
                        -0.368862 -3.15643 6.22113 -0.41633 -3.32351 ...
                '/>
              <Normal vector='0.181806 -0.43043 0.884125 0.219547 -0.395499 0.891841 0.219547 -0.395499
                        0.891841 0.256694 -0.359498 0.897145 0.445922 -0.624509 0.641204 0.445918
                        -0.624508 0.641208 0.468027 -0.541091 0.698692 ...
                '/>
            </IndexedTriangleSet>
          </Shape>
          <Shape>
            ...
        </Group>
      </Scene>
    </X3D>
  </body>
</html>
```

Document Structure

RST Structuring `Gannet.RST.struct.xhtml`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <title>RST Structure</title>
  </head>
  <body>
    <ul id="struct">
      <li xlink:type="extended">
        <!-- 1-19 -->
        <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
        <span id="node_1" xlink:type="resource" xlink:label="parent"></span>
        <span id="edge_1.1" xlink:type="locator" xlink:label="child" xlink:href="#node_2"></span>
      </li>
      <li xlink:type="extended">
        <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
        <span id="node_2" xlink:type="locator" xlink:label="parent"
              xlink:href="Gannet.basic_units.xhtml#u1"></span>
        <span id="edge_2.1" xlink:type="locator" xlink:label="child" xlink:href="#node_3"></span>
      </li>
      <li xlink:type="extended">
        <!-- 2-19 -->
        <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
        <span id="node_3" xlink:type="resource" xlink:label="parent"></span>
        <span id="edge_3.1" xlink:type="locator" xlink:label="child" xlink:href="#node_4"></span>
      </li>
      <li xlink:type="extended">
        <!-- 3-19 -->
        <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
        <span id="node_4" xlink:type="resource" xlink:label="parent"></span>
        <span id="edge_4.1" xlink:type="locator" xlink:label="child"
              xlink:href="Gannet.basic_units.xhtml#u2"></span>
        <span id="edge_4.2" xlink:type="locator" xlink:label="child" xlink:href="#node_5"></span>
      </li>
      <li xlink:type="extended">
        <!-- 4-19 -->
        <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
        <span id="node_5" xlink:type="resource" xlink:label="parent"></span>
        <span id="edge_5.1" xlink:type="locator" xlink:label="child"
              xlink:href="Gannet.basic_units.xhtml#u3"></span>
        <span id="edge_5.2" xlink:type="locator" xlink:label="child" xlink:href="#node_6"></span>
      </li>
      <li xlink:type="extended">
        <!-- 4-12 -->
        <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
```

```
43          <span id="node_6" xlink:type="resource" xlink:label="parent"></span>
44          <span id="edge_6.1" xlink:type="locator" xlink:label="child" xlink:href="#node_7"></span>
45          <span id="edge_6.2" xlink:type="locator" xlink:label="child"
                  xlink:href="Gannet.basic_units.xhtml#u1"></span>
46          <span id="edge_6.3" xlink:type="locator" xlink:label="child" xlink:href="#node_15"></span>
47        </li>
48        <li xlink:type="extended">
49          <!-- 4-11 -->
50          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
51          <span id="node_7" xlink:type="resource" xlink:label="parent"></span>
52          <span id="edge_7.1" xlink:type="locator" xlink:label="child" xlink:href="#node_8"></span>
53          <span id="edge_7.2" xlink:type="locator" xlink:label="child"
                  xlink:href="Gannet.basic_units.xhtml#u9"></span>
54          <span id="edge_7.3" xlink:type="locator" xlink:label="child" xlink:href="#node_12"></span>
55        </li>
56        <li xlink:type="extended">
57          <!-- 4-6 -->
58          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
59          <span id="node_8" xlink:type="resource" xlink:label="parent"></span>
60          <span id="edge_8.1" xlink:type="locator" xlink:label="child" xlink:href="#node_9"></span>
61        </li>
62        <li xlink:type="extended">
63          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
64          <span id="node_9" xlink:type="locator" xlink:label="parent"
                  xlink:href="Gannet.basic_units.xhtml#u6"></span>
65          <span id="edge_9.1" xlink:type="locator" xlink:label="child" xlink:href="#node_10"></span>
66        </li>
67        <li xlink:type="extended">
68          <!-- 5-6 -->
69          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
70          <span id="node_10" xlink:type="resource" xlink:label="parent"></span>
71          <span id="edge_10.1" xlink:type="locator" xlink:label="child" xlink:href="#node_11"></span>
72        </li>
73        <li xlink:type="extended">
74          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
75          <span id="node_11" xlink:type="locator" xlink:label="parent"
                  xlink:href="Gannet.basic_units.xhtml#u7"></span>
76          <span id="edge_11.1" xlink:type="locator" xlink:label="child"
                  xlink:href="Gannet.basic_units.xhtml#u8"></span>
77        </li>
78        <li xlink:type="extended">
79          <!-- 8-11 -->
80          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
81          <span id="node_12" xlink:type="resource" xlink:label="parent"></span>
82          <span id="edge_12.1" xlink:type="locator" xlink:label="child" xlink:href="#node_13"></span>
83          <span id="edge_12.2" xlink:type="locator" xlink:label="child"
                  xlink:href="Gannet.basic_units.xhtml#u12"></span>
84          <span id="edge_12.3" xlink:type="locator" xlink:label="child"
                  xlink:href="Gannet.basic_units.xhtml#u13"></span>
```

```
85        </li>
86        <li xlink:type="extended">
87          <!-- 8-9 -->
88          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
89          <span id="node_13" xlink:type="resource" xlink:label="parent"></span>
90          <span id="edge_13.1" xlink:type="locator" xlink:label="child" xlink:href="#node_14"></span>
91        </li>
92        <li xlink:type="extended">
93          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
94          <span id="node_14" xlink:type="locator" xlink:label="parent"
                xlink:href="Gannet.basic_units.xhtml#u10"></span>
95          <span id="edge_14.1" xlink:type="locator" xlink:label="child"
                xlink:href="Gannet.basic_units.xhtml#u11"></span>
96        </li>
97        <li xlink:type="extended">
98          <!-- 13-19 -->
99          <span xlink:type="arc" xlink:from="parent" xlink:to="child" xlink:show="embed"
                xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"></span>
100         <span id="node_15" xlink:type="resource" xlink:label="parent"></span>
101         <span id="edge_15.1" xlink:type="locator" xlink:label="child"
                xlink:href="Gannet.basic_units.xhtml#u14"></span>
102         <span id="edge_15.2" xlink:type="locator" xlink:label="child"
                xlink:href="Gannet.basic_units.xhtml#u15"></span>
103         <span id="edge_15.3" xlink:type="locator" xlink:label="child"
                xlink:href="Gannet.basic_units.xhtml#u16"></span>
104         <span id="edge_15.4" xlink:type="locator" xlink:label="child"
                xlink:href="Gannet.basic_units.xhtml#u17"></span>
105         <span id="edge_15.5" xlink:type="locator" xlink:label="child"
                xlink:href="Gannet.basic_units.xhtml#u18"></span>
106         <span id="edge_15.6" xlink:type="locator" xlink:label="child"
                xlink:href="Gannet.basic_units.xhtml#u19"></span>
107         <span id="edge_15.7" xlink:type="locator" xlink:label="child"
                xlink:href="Gannet.basic_units.xhtml#u20"></span>
108       </li>
109     </ul>
110   </body>
111 </html>
```

RST Labeling `Gannet.RST.labels.xhtml`

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <title>RST Labels</title>
  </head>
  <body>
    <ol xml:base="Gannet.RST.struct.xhtml" id="labels">
      <li xlink:type="extended">
        <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
        <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
        <span xlink:type="locator" xlink:label="attr" xlink:href="#node_1"
              xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
        <span xlink:type="resource" xlink:label="type"
              xlink:role="http://www.w3.org/2001/XMLSchema#QName">xmlns:rst</span>
        <span xlink:type="resource" xlink:label="value"
              xlink:role="http://www.w3.org/2001/XMLSchema#string">http://www.sfu.ca/rst</span>
      </li>
      <li xlink:type="extended">
        <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
              xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_1"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_3"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_4"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_5"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_6"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_7"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_8"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_10"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_12"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_13"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="locator" xlink:label="elem" xlink:href="#node_15"
              xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
        <span xlink:type="resource" xlink:label="type"
              xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:group</span>
      </li>
      <li xlink:type="extended">
```

```
32          <span xlink:type="arc" xlink:from="elem" xlink:to="type" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
33          <span xlink:type="locator" xlink:label="elem" xlink:href="#node_2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
34          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_4.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
35          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_5.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
36          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_6.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
37          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_7.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
38          <span xlink:type="locator" xlink:label="elem" xlink:href="#node_9"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
39          <span xlink:type="locator" xlink:label="elem" xlink:href="#node_11"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
40          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_11.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
41          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_12.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
42          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_12.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
43          <span xlink:type="locator" xlink:label="elem" xlink:href="#node_14"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
44          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_14.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
45          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_15.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
46          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_15.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
47          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_15.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
48          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_15.4"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
49          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_15.5"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
50          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_15.6"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
51          <span xlink:type="locator" xlink:label="elem" xlink:href="#edge_15.7"
                  xlink:role="http://www.w3.org/2001/XMLSchema#element"></span>
52          <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:segment</span>
53        </li>
54        <li xlink:type="extended">
55          <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
56          <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
57          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_4.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
58          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_6.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
59          <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:relname</span>
```

```
60          <span xlink:type="resource" xlink:label="value"
                  xlink:role="http://www.w3.org/2001/XMLSchema#string">background</span>
61        </li>
62        <li xlink:type="extended">
63          <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
64          <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
65          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_9.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
66          <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:relname</span>
67          <span xlink:type="resource" xlink:label="value"
                  xlink:role="http://www.w3.org/2001/XMLSchema#string">concession</span>
68        </li>
69        <li xlink:type="extended">
70          <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
71          <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
72          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_2.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
73          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_5.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
74          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_11.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
75          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_14.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
76          <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:relname</span>
77          <span xlink:type="resource" xlink:label="value"
                  xlink:role="http://www.w3.org/2001/XMLSchema#string">elaboration</span>
78        </li>
79        <li xlink:type="extended">
80          <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
81          <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
82          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_6.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
83          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_6.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
84          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_7.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
85          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_7.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
86          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_7.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
87          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
88          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
89          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
```

```
90          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.4"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
91          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.5"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
92          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.6"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
93          <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.7"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
94          <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:relname</span>
95          <span xlink:type="resource" xlink:label="value"
                  xlink:role="http://www.w3.org/2001/XMLSchema#string">joint</span>
96        </li>
97        <li xlink:type="extended">
98          <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
99          <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
100         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_12.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
101         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_12.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
102         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_12.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
103         <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:relname</span>
104         <span xlink:type="resource" xlink:label="value"
                  xlink:role="http://www.w3.org/2001/XMLSchema#string">sequence</span>
105       </li>
106       <li xlink:type="extended">
107         <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
108         <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
109         <!-- background -->
110         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_4.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
111         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_6.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
112         <!-- concession -->
113         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_9.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
114         <!-- elaboration -->
115         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_2.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
116         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_5.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
117         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_11.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
118         <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_14.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
119         <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:reltype</span>
120         <span xlink:type="resource" xlink:label="value"
                  xlink:role="http://www.w3.org/2001/XMLSchema#string">rst</span>
```

```
121         </li>
122         <li xlink:type="extended">
123           <span xlink:type="arc" xlink:from="attr" xlink:to="type" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"></span>
124           <span xlink:type="arc" xlink:from="attr" xlink:to="value" xlink:show="embed"
                  xlink:arcrole="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"></span>
125           <!-- joint -->
126           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_6.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
127           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_6.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
128           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_7.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
129           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_7.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
130           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_7.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
131           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
132           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
133           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
134           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.4"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
135           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.5"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
136           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.6"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
137           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_15.7"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
138           <!-- sequence -->
139           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_12.1"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
140           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_12.2"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
141           <span xlink:type="locator" xlink:label="attr" xlink:href="#edge_12.3"
                  xlink:role="http://www.w3.org/2001/XMLSchema#attribute"></span>
142           <span xlink:type="resource" xlink:label="type"
                  xlink:role="http://www.w3.org/2001/XMLSchema#QName">rst:reltype</span>
143           <span xlink:type="resource" xlink:label="value"
                  xlink:role="http://www.w3.org/2001/XMLSchema#string">multinuc</span>
144         </li>
145       </ol>
146     </body>
147   </html>
```

Multistructured Document Description

Gannet.doc.xhtml

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//BUW-DMT//DTD XHTML 1.1 plus XLink 1.1//EN" "xhtml-xlink.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <title>Document</title>
  </head>
  <body>
    <div xlink:type="extended">
      <!-- Structure: RST -->
      <span xlink:type="arc" xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
            xlink:show="embed" xlink:from="struct.RST" xlink:to="labels.RST"></span>
      <span xlink:type="locator" xlink:label="struct.RST" xlink:title="RST Structure"
            xlink:href="Gannet.RST.struct.xhtml#node_1"></span>
      <span xlink:type="locator" xlink:label="labels.RST" xlink:title="RST Labels"
            xlink:href="Gannet.RST.labels.xhtml#labels" xlink:role="http://www.sfu.ca/rst"></span>
      <!-- Content Base -->
      <span xlink:type="arc" xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
            xlink:show="none" xlink:from="content" xlink:to="units"></span>
      <span xlink:type="locator" xlink:label="units" xlink:title="Basic Units"
            xlink:href="Gannet.basic_units.xhtml#basic-units"></span>
      <span xlink:type="locator" xlink:label="content" xlink:title="Text"
            xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod"
            xlink:href="Gannet.content.text.xhtml"></span>
      <span xlink:type="locator" xlink:label="content" xlink:title="List"
            xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod"
            xlink:href="Gannet.content.list.xhtml"></span>
      <span xlink:type="locator" xlink:label="content" xlink:title="Table"
            xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod"
            xlink:href="Gannet.content.table.xhtml"></span>
      <span xlink:type="locator" xlink:label="content" xlink:title="Image"
            xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod"
            xlink:href="Gannet.content.img.xhtml"></span>
      <span xlink:type="locator" xlink:label="content" xlink:title="Audio"
            xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod"
            xlink:href="Gannet.content.audio.xhtml"></span>
      <span xlink:type="locator" xlink:label="content" xlink:title="Video"
            xlink:role="http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod"
            xlink:href="Gannet.content.video.xhtml"></span>
      <span xlink:type="locator" xlink:label="content" xlink:title="3D"
            xlink:role="http://www.web3d.org/specifications/x3d-namespace"
            xlink:href="Gannet.content.3d.xhtml"></span>
    </div>
  </body>
</html>
```

B.2.2. The “Fieldguide” Genre

Content Schema

`Fieldguide.content.sch`

```
<?xml version="1.0" encoding="UTF-8"?>
<schema queryBinding="xslt2" xmlns="http://purl.oclc.org/dsdl/schematron"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:include href="transform/rst/rst-functions.xsl"/>
  <include href="transform/rst/rst.sch#gen"/>

  <ns uri="http://dmt.uni-wuppertal.de/schlupko/xlink4dags" prefix="dag"/>
  <ns uri="http://www.sfu.ca/rst" prefix="rst"/>

  <let name="genre" value="'fieldguide-entry'"/>

  <pattern>
    <title>Fieldguide Entry Genre - Content Specifics</title>
    <rule context="/rst:group">
      <assert test="(count(*) = 1) and rst:segment[rst:is_nucleus(.)]">
        A "<value-of select="$genre"/>"'s top-level group is composed of exactly one content segment
            nucleus.</assert>
    </rule>
    <rule context="/rst:group/rst:segment">
      <assert test="(count(*) = 1) and rst:group[rst:is_satellite(.,'elaboration')]">
        A "<value-of select="$genre"/>"'s initial content segment has exactly one exclusive "rst:group"
            satellite in "<value-of select="rst:*/@relname"/>" relation.</assert>
    </rule>
    <rule context="/rst:group/rst:segment/rst:group">
      <assert test="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]">
        Traversing the "<value-of select="$genre"/>"'s initial satellite's path of descending nuclei leads
            to a "rst:group" consisting of two multinuclear nodes that are in "joint" relation and that
            has a satellite that is in "background" relation to its nucleus.</assert>
      <report test="rst:get_chain( ., rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/parent::rst:*
          )[rst:has_following_satellite(.)]">
        The traversed nuclei starting from the "<value-of select="$genre"/>"'s initial satellite do not
            have following satellites.</report>
    </rule>
  </pattern>

  <xsl:function name="rst:core-structure" as="node()*">
    <xsl:param name="node" as="node()*"/>
    <xsl:sequence select="$node/self::rst:*[
      count(*) = 3 and
      rst:is_multinuclear(*[1],'joint') and
      rst:is_multinuclear(*[2],'joint') and
      rst:is_satellite(*[3],'background')
    ]"/>
  </xsl:function>

</schema>
```

Print-specific Pipeline

`Fieldguide.content2form.print.xpl`

```
<?xml version="1.0" encoding="UTF-8"?>
<p:declare-step version="1.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:c="http://www.w3.org/ns/xproc-step"
    xmlns:cxf="http://xmlcalabash.com/ns/extensions/fileutils"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
    xmlns:tex="http://getfo.sourceforge.net/texml/ns1">

    <p:option name="input-file"  select="'Gannet.doc.xhtml'"/>
    <p:option name="output-file" select="'Gannet.doc.print.pdf'"/>

    <p:option name="texml-exec"  select="'C:/development/libraries/texml-2.0.2/scripts/texml_local.bat'"/>
    <p:option name="latex-exec"  select="'C:/texlive/2014/bin/win32/latex'"/>

    <p:option name="tmp-dir"  select="'transform/TeXML/_tmp/'"/>
    <p:option name="tmp-file" select="'tmp'"/>

    <p:import href="http://xmlcalabash.com/extension/steps/library-1.0.xpl"/>
    <p:import href="transform/stdoff/xlink-validate.xpl"/>
    <p:import href="transform/TeXML/texml2pdf.xpl"/>

    <p:variable name="tmp-path-uri" select="resolve-uri($tmp-dir)"/>
    <p:variable name="tmp-path"     select="if (contains($tmp-path-uri,'file:/')) then
        substring-after($tmp-path-uri,'file:/') else $tmp-path-uri"/>

    <p:load name="dag-input">
        <p:with-option name="href" select="$input-file"/>
    </p:load>

    <dag:validate-with-schematron>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="schema">
            <p:document href="Fieldguide.content.sch"/>
        </p:input>
    </dag:validate-with-schematron>

    <p:sink/>

    <p:xslt>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="stylesheet">
            <p:document href="Fieldguide.content2form.print.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
```

```
    </p:xslt>

    <p:group>

        <p:xslt name="html2texml">
            <p:input port="stylesheet">
                <p:document href="Fieldguide.form2texml.print.xsl"/>
            </p:input>
            <p:input port="parameters">
                <p:empty/>
            </p:input>
        </p:xslt>

        <tex:texml-to-pdf>
            <p:with-option name="texml-exec" select="$texml-exec"/>
            <p:with-option name="latex-exec" select="$latex-exec"/>
            <p:with-option name="file-path" select="$tmp-path"/>
            <p:with-option name="file-name" select="$tmp-file"/>
        </tex:texml-to-pdf>

        <cxf:copy>
            <p:with-option name="href" select="concat($tmp-path-uri,$tmp-file,'.pdf')"/>
            <p:with-option name="target" select="$output-file"/>
        </cxf:copy>

    </p:group>

</p:declare-step>
```

Print-specific Transformation

`Fieldguide.content2form.print.xsl`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:rst="http://www.sfu.ca/rst"
    xmlns:html="http://www.w3.org/1999/xhtml"
    exclude-result-prefixes="#all">

    <xsl:import href="transform/stdoff/stdoff-transform.xsl"/>
    <xsl:import href="transform/rst/rst-functions.xsl"/>

    <xsl:output doctype-public="-//W3C//DTD XHTML 1.1//EN"
        doctype-system="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" indent="yes"/>

    <xsl:template match="/rst:group">
        <xsl:element name="html:html">
            <xsl:apply-templates>
                <!-- set default content mode -->
                <xsl:with-param name="content-mode"
                    select="'http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod'" tunnel="yes"/>
```

```
18              </xsl:apply-templates>
19          </xsl:element>
20      </xsl:template>
21
22      <xsl:template match="/rst:group/rst:segment">
23          <xsl:element name="html:head">
24              <xsl:element name="html:title">
25                  <xsl:call-template name="content-nodes">
26                      <xsl:with-param name="content-mode"
                            select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
27                  </xsl:call-template>
28              </xsl:element>
29          </xsl:element>
30          <xsl:element name="html:body">
31              <xsl:element name="html:h1">
32                  <xsl:call-template name="content-nodes">
33                      <xsl:with-param name="content-mode"
                            select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
34                  </xsl:call-template>
35              </xsl:element>
36              <xsl:call-template name="structure-nodes"/>
37          </xsl:element>
38      </xsl:template>
39
40      <xsl:template match="/rst:group/rst:segment/rst:group">
41          <xsl:element name="html:p">
42              <xsl:apply-templates select="rst:get_preceding_satellite( rst:get_chain( .,
                    rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/parent::rst:* ) )">
43                  <xsl:with-param name="wrapper" select="'html:em'" tunnel="yes"/>
44                  <xsl:with-param name="wrapper-namespace" select="'http://www.w3.org/1999/xhtml'"
                        tunnel="yes"/>
45                  <xsl:with-param name="insert-after" tunnel="yes"><xsl:element
                        name="html:br"/></xsl:with-param>
46                  <xsl:with-param name="content-mode"
                        select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
47              </xsl:apply-templates>
48          </xsl:element>
49          <xsl:element name="html:p">
50              <xsl:apply-templates select="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/*[1]">
51                  <xsl:with-param name="content-mode"
                        select="'http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod'" tunnel="yes"/>
52              </xsl:apply-templates>
53          </xsl:element>
54          <xsl:element name="html:figure">
55              <xsl:apply-templates select="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/*[2]">
56                  <xsl:with-param name="content-mode"
                        select="'http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod'" tunnel="yes"/>
57              </xsl:apply-templates>
58          </xsl:element>
59          <xsl:element name="html:table">
60              <xsl:apply-templates select="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/*[3]">
61                  <xsl:with-param name="wrapper" select="'html:tr'" tunnel="yes"/>
62                  <xsl:with-param name="wrapper-namespace" select="'http://www.w3.org/1999/xhtml'"
                        tunnel="yes"/>
```

```
                <xsl:with-param name="content-mode"
                    select="'http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod'" tunnel="yes"/>
            </xsl:apply-templates>
        </xsl:element>
    </xsl:template>

    <xsl:function name="rst:core-structure" as="node()*">
        <xsl:param name="node" as="node()*"/>
        <xsl:sequence select="$node/self::rst:*[
            count(*) = 3 and
            rst:is_multinuclear(*[1],'joint') and
            rst:is_multinuclear(*[2],'joint') and
            rst:is_satellite(*[3],'background')
        ]"/>
    </xsl:function>

</xsl:stylesheet>
```

Print-specific Output

`Gannet.doc.print.xhtml`

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Gannet</title>
  </head>
  <body>
    <h1>Gannet</h1>
    <p>
      <em>Sula bassana</em>
      <br/>
      <em>Family Sulidae</em>
      <br/>
    </p>
    <p>Birds of the open ocean, Gannets breed on small islands off the <abbr title="North-West">NW</abbr>
       coast of Europe. They move away from land after nesting to winter at sea. The young migrate south
       as far as <abbr title="West">W</abbr> Africa. Gannets feed on fish by plunge-diving from 25<abbr
       title="meters">m</abbr>. They nest in large, noisy colonies. The nest is a pile of seaweed. A
       single egg is incubated for 44 days. The young bird is fed by both parents and flies after 90 days.
       </p>
    <figure>
      <img alt="Adults at breeding colony" src="media/images/gannet.jpg"/>
    </figure>
    <table>
      <tr>
        <th>Size</th>
        <td>Larger than any gull</td>
      </tr>
      <tr>
        <th>Adult</th>
        <td>White, black wing-tips, yellow nape</td>
      </tr>
```

```
      <tr>
        <th>Juvenile</th>
        <td>Grey, gradually becoming white over 5 years</td>
      </tr>
      <tr>
        <th>Bill</th>
        <td>Dagger-like</td>
      </tr>
      <tr>
        <th>In flight</th>
        <td>Cigar-shaped with long, narrow, black-tipped wings</td>
      </tr>
      <tr>
        <th>Voice</th>
        <td>Usually silent, growling <q>urr</q> when nesting</td>
      </tr>
      <tr>
        <th>Lookalikes</th>
        <td>Skuas, Gulls and Terns</td>
      </tr>
    </table>
  </body>
</html>
```

Movie-specific Pipeline

`Fieldguide.content2form.movie.xpl`

```
<?xml version="1.0" encoding="UTF-8"?>
<p:declare-step version="1.0"
    xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:c="http://www.w3.org/ns/xproc-step"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags">

    <p:option name="input-file"  select="'Gannet.doc.xhtml'"/>
    <p:option name="output-file" select="'Gannet.doc.movie.html'"/>

    <p:import href="transform/stdoff/xlink-validate.xpl"/>

    <p:load name="dag-input">
        <p:with-option name="href" select="$input-file"/>
    </p:load>

    <dag:validate-with-schematron>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="schema">
            <p:document href="Fieldguide.content.sch"/>
        </p:input>
    </dag:validate-with-schematron>
    <p:sink/>
```

```
25
26      <p:xslt>
27          <p:input port="source">
28              <p:pipe port="result" step="dag-input"/>
29          </p:input>
30          <p:input port="stylesheet">
31              <p:document href="Fieldguide.content2form.movie.xsl"/>
32          </p:input>
33          <p:input port="parameters">
34              <p:empty/>
35          </p:input>
36      </p:xslt>
37
38      <p:xslt>
39          <p:input port="stylesheet">
40              <p:document href="transform/stdoff/xml-cleanup.xsl"/>
41          </p:input>
42          <p:input port="parameters">
43              <p:empty/>
44          </p:input>
45      </p:xslt>
46
47      <p:store>
48          <p:with-option name="href" select="$output-file"/>
49      </p:store>
50
51  </p:declare-step>
```

Movie-specific Transformation

`Fieldguide.content2form.movie.xsl`

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <xsl:stylesheet version="2.0"
3       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4       xmlns:rst="http://www.sfu.ca/rst"
5       xmlns:html="http://www.w3.org/1999/xhtml"
6       exclude-result-prefixes="#all">
7
8       <xsl:import href="transform/stdoff/stdoff-transform.xsl"/>
9       <xsl:import href="transform/rst/rst-functions.xsl"/>
10
11      <xsl:output doctype-public="-//W3C//DTD XHTML 1.1//EN"
            doctype-system="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" indent="yes"/>
12
13      <xsl:template match="/rst:group">
14
15          <xsl:element name="html:html">
16              <xsl:apply-templates>
17                  <!-- set default content mode -->
18                  <xsl:with-param name="content-mode"
                        select="'http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod'" tunnel="yes"/>
```

```
19              </xsl:apply-templates>
20          </xsl:element>
21
22      </xsl:template>
23
24
25      <xsl:template match="/rst:group/rst:segment">
26
27          <xsl:element name="html:head">
28
29              <xsl:element name="html:title">
30                  <xsl:call-template name="content-nodes">
31                      <xsl:with-param name="content-mode"
                            select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
32                  </xsl:call-template>
33              </xsl:element>
34
35              <xsl:call-template name="media-sync"/>
36
37          </xsl:element>
38
39          <xsl:element name="html:body">
40              <xsl:attribute name="html:onload" select="'javascript:media_seq();'"/>
41
42              <xsl:element name="html:figure">
43
44                  <xsl:element name="html:figcaption">
45                      <xsl:call-template name="content-nodes">
46                          <xsl:with-param name="content-mode"
                                select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
47                      </xsl:call-template>
48                  </xsl:element>
49
50                  <xsl:call-template name="structure-nodes"/>
51
52              </xsl:element>
53
54          </xsl:element>
55
56      </xsl:template>
57
58
59      <xsl:template match="/rst:group/rst:segment/rst:group">
60
61          <xsl:variable name="video">
62              <xsl:apply-templates select="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/*[1]">
63                  <xsl:with-param name="content-mode"
                        select="'http://www.w3.org/MarkUp/DTD/xhtml-video-5.mod'" tunnel="yes"/>
64              </xsl:apply-templates>
65          </xsl:variable>
66          <xsl:element name="html:div">
67              <xsl:attribute name="data-timecontainer" select="'seq'"/>
68              <xsl:apply-templates select="$video/node()">
69                  <xsl:with-param name="thumbnail" tunnel="yes">
```

```
70                      <xsl:apply-templates
                            select="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/*[2]">
71                          <xsl:with-param name="content-mode"
                                select="'http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod'" tunnel="yes"/>
72                      </xsl:apply-templates>
73                  </xsl:with-param>
74              </xsl:apply-templates>
75          </xsl:element>
76
77          <xsl:variable name="audio">
78              <xsl:apply-templates select="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/*[1]">
79                  <xsl:with-param name="content-mode"
                        select="'http://www.w3.org/MarkUp/DTD/xhtml-audio-5.mod'" tunnel="yes"/>
80              </xsl:apply-templates>
81          </xsl:variable>
82          <xsl:element name="html:div">
83              <xsl:apply-templates select="$audio/node()">
84                  <xsl:with-param name="video" select="$video//html:video" tunnel="yes"/>
85              </xsl:apply-templates>
86          </xsl:element>
87
88      </xsl:template>
89
90
91      <xsl:template match="html:video">
92          <xsl:param name="thumbnail" tunnel="yes"/>
93
94          <xsl:copy>
95              <xsl:attribute name="preload" select="'auto'"/>
96              <xsl:attribute name="mediagroup" select="generate-id(.)"/>
97              <xsl:if test="position() = 1 and $thumbnail//html:img[1]/@src">
98                  <xsl:attribute name="poster" select="$thumbnail//html:img[1]/@src"/>
99              </xsl:if>
100             <xsl:apply-templates select="node()|@*[name() != 'controls']" mode="process-dag"/>
101         </xsl:copy>
102
103     </xsl:template>
104
105
106     <xsl:template match="html:audio">
107         <xsl:param name="video" tunnel="yes"/>
108         <xsl:variable name="pos" select="position()"/>
109
110         <xsl:copy>
111             <xsl:attribute name="preload" select="'auto'"/>
112             <xsl:attribute name="mediagroup" select="generate-id($video[$pos])"/>
113             <xsl:apply-templates select="node()|@*[name() != 'controls']" mode="process-dag"/>
114         </xsl:copy>
115
116     </xsl:template>
117
118
119     <xsl:function name="rst:core-structure" as="node()*">
120         <xsl:param name="node" as="node()*"/>
121         <xsl:sequence select="$node/self::rst:*[
```

```
            count(*) = 3 and
            rst:is_multinuclear(*[1],'joint') and
            rst:is_multinuclear(*[2],'joint') and
            rst:is_satellite(*[3],'background')
        ]"/>
    </xsl:function>


    <xsl:template name="media-sync">

        <xsl:element name="html:style">
            <xsl:attribute name="html:type" select="'text/css'"/>
            <xsl:comment><![CDATA[
            figure {position:relative; margin:0;}
            figure > * {position:absolute;}
            video {height:346px; width:464px; background-color:#000;}
            figcaption {
                    z-index:1;
                    margin:0;
                    padding:.2em .5em;
                    font-size:2em;
                    font-weight:bold;
                    font-family:Calibri, Candara, Segoe, "Segoe UI", Optima, Arial, sans-serif;
                    color:white;
                    background-color:rgba(0,0,0,.5);
            }]]></xsl:comment>
        </xsl:element>

        <xsl:element name="html:script">
            <xsl:attribute name="html:type" select="'text/javascript'"/>
            <xsl:comment><![CDATA[
            function SyncGroup(node){
                var id = node.getAttribute('mediagroup');
                var nodes = document.body.querySelectorAll('[mediagroup="' + id + '"]');

                this.play = function () {
                    for(var i = 0; i < nodes.length; i++) {
                        nodes[i].play();
                    }
                }
                this.pause = function () {
                    for(var i = 0; i < nodes.length; i++) {
                        nodes[i].pause();
                    }
                }
                this.ended = function () {
                    for(var i = 0; i < nodes.length; i++) {
                        if (nodes[i] != node) {
                            nodes[i].pause();
                        }
                    }
                }
                this.timeupdate = function () {
                    for(var i = 0; i < nodes.length; i++) {
                        if (nodes[i] != node) {
```

```
                            nodes[i].currentTime = node.currentTime;
                        }
                    }
                }
            }

            function sync_mediagroup(node){
                var group = new SyncGroup(node);
                node.addEventListener("play",group.play,false);
                node.addEventListener("pause",group.pause,false);
                node.addEventListener("ended",group.ended,false);
                node.addEventListener("seeked",group.timeupdate,false);
            }

            function media_seq() {
                var container = document.querySelector('[data-timecontainer=seq]');
                container.style.position = "relative";
                var media = container.querySelectorAll('video,audio');
                for(var i = 0; i < media.length; i++) {
                    sync_mediagroup(media[i]);
                    media[i].style.position = "absolute";
                    media[i].style.zIndex = -i;
                    media[i].load();
                    media[i].volume=0.25;

                    media[i].addEventListener('mouseover', function() {
                        this.setAttribute("controls","controls"); }, false);
                    media[i].addEventListener('mouseout', function() { this.removeAttribute("controls");
                        }, false);

                    media[i].onended = function(){
                        if (this.nextElementSibling) this.nextElementSibling.play();
                        this.style.zIndex = this.style.zIndex - media.length;
                        this.load();
                    };
                }
            }
        //]]></xsl:comment>
        </xsl:element>

    </xsl:template>

</xsl:stylesheet>
```

Movie-specific Output

`Gannet.doc.movie.html`

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Gannet</title>
    <style type="text/css"> ...
```

```
26      </style>
27      <script type="text/javascript"> ...
92      </script>
93    </head>
94    <body onload="javascript:media_seq();">
95      <figure>
96        <figcaption>Gannet</figcaption>
97        <div data-timecontainer="seq">
98          <video preload="auto" mediagroup="d504e1" poster="media/images/gannet.jpg" width="480"
              height="246">
99            <source src="media/video/6.mp4" type="video/mp4"></source>
100         </video>
101         <video preload="auto" mediagroup="d504e3" width="480" height="246">
102           <source src="media/video/7.mp4" type="video/mp4"></source>
103         </video>
104         <video preload="auto" mediagroup="d504e5" width="480" height="246">
105           <source src="media/video/8.mp4" type="video/mp4"></source>
106         </video>
107         <video preload="auto" mediagroup="d504e7" width="480" height="246">
108           <source src="media/video/9.mp4" type="video/mp4"></source>
109         </video>
110         <video preload="auto" mediagroup="d504e9" width="480" height="246">
111           <source src="media/video/10.mp4" type="video/mp4"></source>
112         </video>
113         <video preload="auto" mediagroup="d504e11" width="480" height="246">
114           <source src="media/video/11.mp4" type="video/mp4"></source>
115         </video>
116         <video preload="auto" mediagroup="d504e13" width="480" height="246">
117           <source src="media/video/12.mp4" type="video/mp4"></source>
118         </video>
119         <video preload="auto" mediagroup="d504e15" width="480" height="246">
120           <source src="media/video/13.mp4" type="video/mp4"></source>
121         </video>
122       </div>
123       <div>
124         <audio preload="auto" mediagroup="d504e1">
125           <source src="media/audio/6.wav" type="audio/wav"></source>
126         </audio>
127         <audio preload="auto" mediagroup="d504e3">
128           <source src="media/audio/7.wav" type="audio/wav"></source>
129         </audio>
130         <audio preload="auto" mediagroup="d504e5">
131           <source src="media/audio/8.wav" type="audio/wav"></source>
132         </audio>
133         <audio preload="auto" mediagroup="d504e7">
134           <source src="media/audio/9.wav" type="audio/wav"></source>
135         </audio>
136         <audio preload="auto" mediagroup="d504e9">
137           <source src="media/audio/10.wav" type="audio/wav"></source>
138         </audio>
139         <audio preload="auto" mediagroup="d504e11">
140           <source src="media/audio/11.wav" type="audio/wav"></source>
141         </audio>
142         <audio preload="auto" mediagroup="d504e13">
143           <source src="media/audio/12.wav" type="audio/wav"></source>
```

```
          </audio>
          <audio preload="auto" mediagroup="d504e15">
            <source src="media/audio/13.wav" type="audio/wav"></source>
          </audio>
        </div>
      </figure>
    </body>
</html>
```

3D-specific Pipeline

`Fieldguide.content2form.3d.xpl`

```
<?xml version="1.0" encoding="UTF-8"?>
<p:declare-step version="1.0"
    xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:c="http://www.w3.org/ns/xproc-step"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags">

    <p:option name="input-file"  select="'Gannet.doc.xhtml'"/>
    <p:option name="output-file" select="'Gannet.doc.3d.html'"/>

    <p:import href="transform/stdoff/xlink-validate.xpl"/>

    <p:load name="dag-input">
        <p:with-option name="href" select="$input-file"/>
    </p:load>

    <dag:validate-with-schematron>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="schema">
            <p:document href="Fieldguide.content.sch"/>
        </p:input>
    </dag:validate-with-schematron>
    <p:sink/>

    <p:xslt name="transform">
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="stylesheet">
            <p:document href="Fieldguide.content2form.3d.xsl"/>
        </p:input>
        <p:with-param name="poster-url" select="$output-file"/>
    </p:xslt>

    <p:for-each>
        <p:iteration-source>
            <p:pipe port="secondary" step="transform"/>
        </p:iteration-source>

        <p:xslt>
            <p:input port="stylesheet">
                <p:document href="transform/stdoff/xml-cleanup.xsl"/>
            </p:input>
            <p:input port="parameters">
                <p:empty/>
            </p:input>
        </p:xslt>

        <p:store>
            <p:with-option name="href" select="concat(
```

```
                if (count(tokenize($output-file,'\.')) &gt; 1)
                    then string-join(tokenize($output-file,'\.')[position() &lt; last()],'.')
                    else $output-file
                ,
                '.',
                p:iteration-position(),
                '.svg'
            )"/>
        </p:store>

    </p:for-each>

    <p:xslt>
        <p:input port="source">
            <p:pipe port="result" step="transform"/>
        </p:input>
        <p:input port="stylesheet">
            <p:document href="transform/stdoff/xml-cleanup.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
    </p:xslt>

    <p:store>
        <p:with-option name="href" select="$output-file"/>
    </p:store>

</p:declare-step>
```

3D-specific Transformation

`Fieldguide.content2form.3d.xsl`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
    xmlns:rst="http://www.sfu.ca/rst"
    xmlns:html="http://www.w3.org/1999/xhtml"
    xmlns:x3d="http://www.web3d.org/specifications/x3d-namespace"
    xmlns:svg="http://www.w3.org/2000/svg"
    exclude-result-prefixes="#all">

    <xsl:import href="transform/stdoff/stdoff-transform.xsl"/>
    <xsl:import href="transform/rst/rst-functions.xsl"/>

    <xsl:param name="poster-url" select="'poster.svg'"/>

    <xsl:template match="/rst:group">

```

```
19          <xsl:element name="html:html">
20              <xsl:apply-templates>
21                  <!-- set default content mode -->
22                  <xsl:with-param name="content-mode"
                        select="'http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod'" tunnel="yes"/>
23              </xsl:apply-templates>
24          </xsl:element>
25
26      </xsl:template>
27
28      <xsl:template match="/rst:group/rst:segment">
29
30          <xsl:element name="html:head">
31
32              <xsl:element name="html:title">
33                  <xsl:call-template name="content-nodes">
34                      <xsl:with-param name="content-mode"
                            select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
35                  </xsl:call-template>
36              </xsl:element>
37
38              <xsl:element name="html:script">
39                  <xsl:attribute name="type" select="'text/javascript'"/>
40                  <xsl:attribute name="src"  select="'script/x3dom-1.7.0/x3dom.js'"/>
41                  <xsl:call-template name="keep-endtag"/>
42              </xsl:element>
43              <xsl:element name="html:script">
44                  <xsl:attribute name="type" select="'text/javascript'"/>
45                  <xsl:attribute name="src"  select="'script/x3dom-1.7.0/components/CADGeometry.js'"/>
46                  <xsl:call-template name="keep-endtag"/>
47              </xsl:element>
48              <xsl:element name="html:link">
49                  <xsl:attribute name="rel"  select="'stylesheet'"/>
50                  <xsl:attribute name="type" select="'text/css'"/>
51                  <xsl:attribute name="href" select="'script/x3dom-1.7.0/x3dom.css'"/>
52              </xsl:element>
53
54          </xsl:element>
55
56          <xsl:element name="html:body">
57
58              <xsl:element name="x3d:X3D">
59                  <xsl:attribute name="width"  select="'800px'"/>
60                  <xsl:attribute name="height" select="'500px'"/>
61
62                  <xsl:element name="x3d:Scene">
63
64                      <xsl:call-template name="structure-nodes"/>
65
66                      <xsl:element name="x3d:Viewpoint">
67                          <xsl:attribute name="centerOfRotation" select="'0 -0.42775 0.461924'"/>
68                          <xsl:attribute name="position" select="'0 -0.42775 4.13119'"/>
69                      </xsl:element>
70                  </xsl:element>
71
```

```
72              </xsl:element>
73
74          </xsl:element>
75
76      </xsl:template>
77
78
79      <xsl:template match="/rst:group/rst:segment/rst:group">
80          <xsl:variable name="poster-width"  select="841"/>
81          <xsl:variable name="poster-height" select="1189"/>
82          <xsl:variable name="poster-name"
                select="rst:substring-before-last(rst:substring-after-last($poster-url,'/'),'\.')"/>
83
84          <!-- left poster -->
85          <xsl:variable name="poster-1" select="concat($poster-name,'.1.svg')"/>
86          <!-- left poster canvas -->
87          <xsl:call-template name="model-poster">
88              <xsl:with-param name="width" select="$poster-width"/>
89              <xsl:with-param name="height" select="$poster-height"/>
90              <xsl:with-param name="poster-url" select="$poster-1"/>
91              <xsl:with-param name="content">
92                  <xsl:element name="html:h1">
93                      <xsl:apply-templates select="dag:content-nodes(/rst:group/rst:segment)">
94                          <xsl:with-param name="content-mode"
                                select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
95                      </xsl:apply-templates>
96                  </xsl:element>
97                  <xsl:element name="html:p">
98                      <xsl:apply-templates select="rst:get_preceding_satellite( rst:get_chain( .,
                            rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/parent::rst:* ) )">
99                          <xsl:with-param name="wrapper" select="'html:em'" tunnel="yes"/>
100                         <xsl:with-param name="wrapper-namespace" select="'http://www.w3.org/1999/xhtml'"
                                tunnel="yes"/>
101                         <xsl:with-param name="insert-after" tunnel="yes"><xsl:element
                                name="html:br"/></xsl:with-param>
102                         <xsl:with-param name="content-mode"
                                select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
103                     </xsl:apply-templates>
104                 </xsl:element>
105                 <xsl:element name="html:ul">
106                     <xsl:apply-templates
                            select="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/*[1]">
107                         <xsl:with-param name="wrapper" select="'html:li'" tunnel="yes"/>
108                         <xsl:with-param name="wrapper-namespace" select="'http://www.w3.org/1999/xhtml'"
                                tunnel="yes"/>
109                         <xsl:with-param name="content-mode"
                                select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
110                     </xsl:apply-templates>
111                 </xsl:element>
112             </xsl:with-param>
113         </xsl:call-template>
114         <!-- left poster frame -->
115         <xsl:element name="x3d:Transform">
116             <xsl:attribute name="translation" select="string-join((string(-.001 * $poster-width), 0, 0),'
                    ')"/>
```

```
117             <xsl:call-template name="model-posterframe">
118                 <xsl:with-param name="width" select="$poster-width"/>
119                 <xsl:with-param name="height" select="$poster-height"/>
120                 <xsl:with-param name="poster-url" select="$poster-1"/>
121             </xsl:call-template>
122         </xsl:element>
123 
124         <!-- right poster -->
125         <xsl:variable name="poster-2" select="concat($poster-name,'.2.svg')"/>
126         <!-- right poster canvas -->
127         <xsl:call-template name="model-poster">
128             <xsl:with-param name="width" select="$poster-width"/>
129             <xsl:with-param name="height" select="$poster-height"/>
130             <xsl:with-param name="poster-url" select="$poster-2"/>
131             <xsl:with-param name="content">
132                 <xsl:element name="html:ul">
133                     <xsl:apply-templates
                            select="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/*[3]">
134                         <xsl:with-param name="wrapper" select="'html:li'" tunnel="yes"/>
135                         <xsl:with-param name="wrapper-namespace" select="'http://www.w3.org/1999/xhtml'"
                                tunnel="yes"/>
136                         <xsl:with-param name="content-mode"
                                select="'http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod'" tunnel="yes"/>
137                     </xsl:apply-templates>
138                 </xsl:element>
139             </xsl:with-param>
140         </xsl:call-template>
141         <!-- right poster frame -->
142         <xsl:element name="x3d:Transform">
143             <xsl:attribute name="translation" select="string-join((string(.001 * $poster-width), 0, 0),'
                    ')"/>
144             <xsl:call-template name="model-posterframe">
145                 <xsl:with-param name="width" select="$poster-width"/>
146                 <xsl:with-param name="height" select="$poster-height"/>
147                 <xsl:with-param name="poster-url" select="$poster-2"/>
148             </xsl:call-template>
149         </xsl:element>
150 
151         <!-- centre display -->
152         <xsl:element name="x3d:Transform">
153             <xsl:attribute name="translation" select="string-join(('0', '-1', '.5'),' ')"/>
154             <xsl:element name="x3d:Shape">
155                 <xsl:element name="x3d:Appearance">
156                     <xsl:element name="x3d:Material">
157                         <xsl:attribute name="diffuseColor" select="string-join(('.8', '.8', '.8'),' ')"/>
158                     </xsl:element>
159                 </xsl:element>
160                 <xsl:element name="x3d:Cylinder">
161                     <xsl:attribute name="height" select="1"/>
162                     <xsl:attribute name="radius" select=".25"/>
163                 </xsl:element>
164             </xsl:element>
165         </xsl:element>
166 
167         <!-- bird -->
```

```
168         <xsl:element name="x3d:Transform">
169             <xsl:attribute name="translation" select="string-join(('0', '-.51', '.5'),' ')"/>
170             <xsl:attribute name="rotation"    select="string-join(('0', '1', '0', '1.57'),' ')"/>
171             <xsl:attribute name="scale"       select="string-join(('.1', '.1', '.1'),' ')"/>
172             <xsl:apply-templates select="rst:traverse_nucleus_chain(.)[rst:core-structure(.)]/*[2]">
173                 <xsl:with-param name="content-mode"
                        select="'http://www.web3d.org/specifications/x3d-namespace'" tunnel="yes"/>
174             </xsl:apply-templates>
175         </xsl:element>
176
177     </xsl:template>
178
179
180     <xsl:template name="model-poster">
181         <xsl:param name="width"   select="841"/>
182         <xsl:param name="height"  select="1189"/>
183         <xsl:param name="content" select="()" as="node()*"/>
184         <xsl:param name="poster-url"/>
185
186         <xsl:result-document href="{$poster-url}">
187             <xsl:element name="svg:svg">
188                 <xsl:attribute name="version" select="'1.1'"/>
189                 <xsl:attribute name="width"   select="$width"/>
190                 <xsl:attribute name="height"  select="$height"/>
191                 <xsl:attribute name="style"   select="'background-color:white'"/>
192
193                 <xsl:element name="svg:switch">
194                     <xsl:element name="svg:foreignObject">
195                         <xsl:attribute name="width"   select="$width"/>
196                         <xsl:attribute name="height"  select="$height"/>
197
198                         <xsl:element name="html:html">
199
200                             <xsl:element name="html:head">
201                                 <xsl:element name="html:title"/>
202                                 <xsl:element name="html:style">
203                                     <xsl:attribute name="type" select="'text/css'"/>
204                                     <xsl:text><![CDATA[
205                                         body {font-size: 200%; margin: 2em;}
206                                         header p {font-style: italic;}
207                                         ul {line-height: 1.7em;}
208                                         abbr {border: none;}
209                                     ]]></xsl:text>
210                                 </xsl:element>
211                             </xsl:element>
212
213                             <xsl:element name="html:body">
214                                 <xsl:copy-of select="$content"/>
215                             </xsl:element>
216
217                         </xsl:element>
218                     </xsl:element>
219                 </xsl:element>
220             </xsl:element>
221         </xsl:result-document>
```

```
222
223     </xsl:template>
224
225     <xsl:template name="model-posterframe">
226         <xsl:param name="width"      select="841"/>
227         <xsl:param name="height"     select="1189"/>
228         <xsl:param name="thickness"  select="50"/>
229         <xsl:param name="color"      select="'.2 .2 .2'"/>
230         <xsl:param name="poster-url"/>
231
232         <xsl:variable name="dwidth"  select="string($width  div 1000)"/>
233         <xsl:variable name="dheight" select="string($height div 1000)"/>
234         <xsl:variable name="dthick"  select="string($thickness div 1000)"/>
235
236         <!-- poster canvas -->
237         <xsl:element name="x3d:Shape">
238             <xsl:element name="x3d:Appearance">
239                 <xsl:element name="x3d:ImageTexture">
240                     <xsl:attribute name="hideChildren" select="'false'"/>
241                     <xsl:attribute name="url" select="$poster-url"/>
242                 </xsl:element>
243             </xsl:element>
244             <xsl:element name="x3d:Box">
245                 <xsl:attribute name="size" select="string-join(($dwidth, $dheight, '0'),' ')"/>
246             </xsl:element>
247         </xsl:element>
248         <!-- left frame fragment -->
249         <xsl:call-template name="model-framefragment">
250             <xsl:with-param name="size"  select="string-join(($dthick, $dheight, $dthick),' ')"/>
251             <xsl:with-param name="trans" select="string-join((string(-1 * (number($dwidth) +
                    number($dthick)) div 2), '0', '0'),' ')"/>
252             <xsl:with-param name="color" select="$color"/>
253         </xsl:call-template>
254         <!-- right frame fragment -->
255         <xsl:call-template name="model-framefragment">
256             <xsl:with-param name="size"  select="string-join(($dthick, $dheight, $dthick),' ')"/>
257             <xsl:with-param name="trans" select="string-join((string((number($dwidth) + number($dthick))
                    div 2), '0', '0'),' ')"/>
258             <xsl:with-param name="color" select="$color"/>
259         </xsl:call-template>
260         <!-- bottom frame fragment -->
261         <xsl:call-template name="model-framefragment">
262             <xsl:with-param name="size"  select="string-join(((string(number($dwidth) + (2 *
                    number($dthick)))), $dthick, $dthick),' ')"/>
263             <xsl:with-param name="trans" select="string-join(('0', string(-1 * (number($dheight) +
                    number($dthick)) div 2), '0'),' ')"/>
264             <xsl:with-param name="color" select="$color"/>
265         </xsl:call-template>
266         <!-- top frame fragment -->
267         <xsl:call-template name="model-framefragment">
268             <xsl:with-param name="size"  select="string-join(((string(number($dwidth) + (2 *
                    number($dthick)))), $dthick, $dthick),' ')"/>
269             <xsl:with-param name="trans" select="string-join(('0', string((number($dheight) +
                    number($dthick)) div 2), '0'),' ')"/>
270             <xsl:with-param name="color" select="$color"/>
```

```
271         </xsl:call-template>
272
273     </xsl:template>
274
275     <xsl:template name="model-framefragment">
276         <xsl:param name="size"/>
277         <xsl:param name="trans"/>
278         <xsl:param name="color"/>
279
280         <xsl:element name="x3d:Transform">
281             <xsl:attribute name="translation" select="$trans"/>
282             <xsl:element name="x3d:Shape">
283                 <xsl:element name="x3d:Appearance">
284                     <xsl:element name="x3d:Material">
285                         <xsl:attribute name="diffuseColor" select="$color"/>
286                     </xsl:element>
287                 </xsl:element>
288                 <xsl:element name="x3d:Box">
289                     <xsl:attribute name="size" select="$size"/>
290                 </xsl:element>
291             </xsl:element>
292         </xsl:element>
293     </xsl:template>
294
295
296     <xsl:template name="keep-endtag">
297         <xsl:text>&#x200B;</xsl:text>
298     </xsl:template>
299
300
301     <xsl:function name="rst:core-structure" as="node()*">
302         <xsl:param name="node" as="node()*"/>
303         <xsl:sequence select="$node/self::rst:*[
304             count(*) = 3 and
305             rst:is_multinuclear(*[1],'joint') and
306             rst:is_multinuclear(*[2],'joint') and
307             rst:is_satellite(*[3],'background')
308         ]"/>
309     </xsl:function>
310
311     <xsl:function name="rst:substring-before-last">
312         <xsl:param name="string"  as="xs:string"/>
313         <xsl:param name="pattern" as="xs:string"/>
314         <xsl:variable name="tokens" select="tokenize($string,$pattern)"/>
315         <xsl:value-of select="if (count($tokens) &gt; 1) then string-join($tokens[position() &lt;
              last()],'.') else $string"/>
316     </xsl:function>
317
318     <xsl:function name="rst:substring-after-last">
319         <xsl:param name="string"  as="xs:string"/>
320         <xsl:param name="pattern" as="xs:string"/>
321         <xsl:variable name="tokens" select="tokenize($string,$pattern)"/>
322         <xsl:value-of select="if (count($tokens) &gt; 1) then $tokens[last()] else $string"/>
323     </xsl:function>
324 </xsl:stylesheet>
```

3D-specific Output

Gannet.doc.3d.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Gannet</title>
    <script type="text/javascript" src="script/x3dom-1.7.0/x3dom.js"></script>
    <script type="text/javascript" src="script/x3dom-1.7.0/components/CADGeometry.js"></script>
    <link rel="stylesheet" type="text/css" href="script/x3dom-1.7.0/x3dom.css"/>
  </head>
  <body>
    <X3D xmlns="http://www.web3d.org/specifications/x3d-namespace" width="800px" height="500px">
      <Scene>
        <Transform translation="-0.841 0 0">
          <Shape>
            <Appearance>
              <ImageTexture hideChildren="false" url="Gannet.doc.3d.1.svg"></ImageTexture>
            </Appearance>
            <Box size="0.841 1.189 0"></Box>
          </Shape>
          <Transform translation="-0.4455 0 0">
            <Shape>
              <Appearance>
                <Material diffuseColor=".2 .2 .2"></Material>
              </Appearance>
              <Box size="0.05 1.189 0.05"></Box>
            </Shape>
          </Transform>
          <Transform translation="0.4455 0 0">
            <Shape>
              <Appearance>
                <Material diffuseColor=".2 .2 .2"></Material>
              </Appearance>
              <Box size="0.05 1.189 0.05"></Box>
            </Shape>
          </Transform>
          <Transform translation="0 -0.6195 0">
            <Shape>
              <Appearance>
                <Material diffuseColor=".2 .2 .2"></Material>
              </Appearance>
              <Box size="0.941 0.05 0.05"></Box>
            </Shape>
          </Transform>
          <Transform translation="0 0.6195 0">
            <Shape>
              <Appearance>
                <Material diffuseColor=".2 .2 .2"></Material>
              </Appearance>
              <Box size="0.941 0.05 0.05"></Box>
            </Shape>
          </Transform>
        </Transform>
        <Transform translation="0.841 0 0">
```

```
52            <Shape>
53              <Appearance>
54                <ImageTexture hideChildren="false" url="Gannet.doc.3d.2.svg"></ImageTexture>
55              </Appearance>
56              <Box size="0.841 1.189 0"></Box>
57            </Shape>
58            <Transform translation="-0.4455 0 0">
59              <Shape>
60                <Appearance>
61                  <Material diffuseColor=".2 .2 .2"></Material>
62                </Appearance>
63                <Box size="0.05 1.189 0.05"></Box>
64              </Shape>
65            </Transform>
66            <Transform translation="0.4455 0 0">
67              <Shape>
68                <Appearance>
69                  <Material diffuseColor=".2 .2 .2"></Material>
70                </Appearance>
71                <Box size="0.05 1.189 0.05"></Box>
72              </Shape>
73            </Transform>
74            <Transform translation="0 -0.6195 0">
75              <Shape>
76                <Appearance>
77                  <Material diffuseColor=".2 .2 .2"></Material>
78                </Appearance>
79                <Box size="0.941 0.05 0.05"></Box>
80              </Shape>
81            </Transform>
82            <Transform translation="0 0.6195 0">
83              <Shape>
84                <Appearance>
85                  <Material diffuseColor=".2 .2 .2"></Material>
86                </Appearance>
87                <Box size="0.941 0.05 0.05"></Box>
88              </Shape>
89            </Transform>
90          </Transform>
91          <Transform translation="0 -1 .5">
92            <Shape>
93              <Appearance>
94                <Material diffuseColor=".8 .8 .8"></Material>
95              </Appearance>
96              <Cylinder height="1" radius="0.25"></Cylinder>
97            </Shape>
98          </Transform>
99          <Transform translation="0 -.51 .5" rotation="0 1 0 1.57" scale=".1 .1 .1">
              <!-- bird model --> ...
215         </Transform>
216         <Viewpoint centerOfRotation="0 -0.42775 0.461924" position="0 -0.42775 4.13119"></Viewpoint>
217       </Scene>
218     </X3D>
219   </body>
220 </html>
```

Gannet.doc.3d.1.svg

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="841" height="1189"
  style="background-color:white">
  <switch>
    <foreignObject width="841" height="1189">
      <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
          <title/>
          <style type="text/css">
            body {font-size: 200%; margin: 2em;}
            header p {font-style: italic;}
            ul {line-height: 1.7em;}
            abbr {border: none;}
          </style>
        </head>
        <body>
          <h1>Gannet</h1>
          <p>
            <em>Sula bassana</em>
            <br/>
            <em>Family Sulidae</em>
            <br/>
          </p>
          <ul>
            <li>Breed on small islands off the <abbr title="North-West">NW</abbr> coast of Europe</li>
            <li>Move away from land after nesting to winter at sea</li>
            <li>Young migrate south as far as <abbr title="West">W</abbr> Africa</li>
            <li>Feed on fish by plunge-diving from 25<abbr title="meters">m</abbr></li>
            <li>Nest in large noisy colonies</li>
            <li>Nest pile of seaweed</li>
            <li>Single egg incubated for 44 days</li>
            <li>Young fed by both parents, flies after 90 days</li>
          </ul>
        </body>
      </html>
    </foreignObject>
  </switch>
</svg>
```

Gannet.doc.3d.2.svg

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="841" height="1189"
  style="background-color:white">
  <switch>
    <foreignObject width="841" height="1189">
      <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
          <title/>
          <style type="text/css">
            body {font-size: 200%; margin: 2em;}
            header p {font-style: italic;}
            ul {line-height: 1.7em;}
            abbr {border: none;}
          </style>
        </head>
        <body>
          <ul>
            <li>Larger than any gull</li>
            <li>Adult white, black wing-tips, yellow nape</li>
            <li>Juvenile grey, gradually becoming white over 5 years</li>
            <li>Bill dagger-like</li>
            <li>In Flight, cigar-shaped with long, narrow, black-tipped wings</li>
            <li>Usually silent, growling <q>urr</q> when nesting</li>
            <li>Lookalikes Skuas, Gulls and Terns</li>
          </ul>
        </body>
      </html>
    </foreignObject>
  </switch>
</svg>
```

C. Implementation

This appendix compiles the technical framework that is developed in the thesis.

Specifically, Appendix C.1 provides the implementation of the multistructured document processing workflow (Sec. 4.2); Appendix C.2 provides the implementation of the RST-specific processing workflows (Sec. 4.3); Appendix C.3 provides the implementation of the genre processing workflow (Sec. 4.3).

C.1. Multistructured Document Processing

C.1.1. XLink Traversal

xlink-traverse.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=============================================================================================================
    XSLT stylesheet to copy documents and process embedded XLink descriptions
    @author   Frederik R.N. Schlupkothen
    @version  15.06.2015
    =============================================================================================================-->
<xsl:stylesheet version="2.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
    exclude-result-prefixes="#all">

    <xsl:import href="dag-functions.xsl"/>


    <xsl:template match="/">
        <xsl:apply-templates mode="process-link"/>
    </xsl:template>


<!--process-link-->

<!---
    Process XLink extended-type elements.
    @param element()   $link          XLink extended-type element (the matched element).
    @param element()?  $link-focus    The originally referenced element (arc or resource) within the link.
    @param xs:boolean  $via-linkbase  Signalizing whether the context node is part of a referenced
          linkbase or not.
-->
    <xsl:template match="*[@xlink:type = 'extended']" name="process-link" mode="process-link">
        <xsl:param name="link"         as="element()"  select="."/>
        <xsl:param name="link-focus"   as="element()?" select="()"/>
        <xsl:param name="via-linkbase" as="xs:boolean" select="false()"/>

        <xsl:variable name="link-arcs" as="element()*">
            <xsl:choose>
                <!-- traverse specific arc -->
                <xsl:when test="$link-focus/@xlink:type = 'arc'">
                    <xsl:call-template name="get-arc-type-elements">
                        <xsl:with-param name="link" select="$link"/>
                        <xsl:with-param name="arcs" select="$link-focus"/>
                    </xsl:call-template>
                </xsl:when>
                <!-- traverse whole link -->
                <xsl:otherwise>
                    <xsl:call-template name="get-arc-type-elements">
                        <xsl:with-param name="link" select="$link"/>
                    </xsl:call-template>
```

```
49                </xsl:otherwise>
50            </xsl:choose>
51        </xsl:variable>
52
53        <xsl:apply-templates select="$link-arcs" mode="process-link">
54            <xsl:with-param name="link" select="$link"/>
55            <xsl:with-param name="link-focus" select="$link-focus"/>
56            <xsl:with-param name="via-linkbase" select="$via-linkbase"/>
57        </xsl:apply-templates>
58
59    </xsl:template>
60
61
62  <!---
63      Process XLink arc-type elements.
64      @param element()   $arc           XLink arc-type element (the matched element).
65      @param element()   $link          XLink extended-type elements (the matched element's parent).
66      @param element()?  $link-focus    The originally referenced element (arc or resource) within the link.
67      @param xs:boolean  $via-linkbase  Signalizing whether the context node is part of a referenced
            linkbase or not.
68  -->
69      <xsl:template match="*[@xlink:type = 'arc']" mode="process-link">
70          <xsl:param name="arc"          as="element()"  select="."/>
71          <xsl:param name="link"         as="element()"  select=".."/>
72          <xsl:param name="link-focus"   as="element()?" select="()"/>
73          <xsl:param name="via-linkbase" as="xs:boolean" select="false()"/>
74
75          <xsl:param name="is-resource-focussed" as="xs:boolean" select="$link-focus/@xlink:type =
                'resource' or $link-focus/@xlink:type = 'locator'"/>
76
77          <xsl:apply-templates select="$link/*[@xlink:label =
                current()/@xlink:from][not($is-resource-focussed) or dag:is_same(.,$link-focus)]"
                mode="process-link">
78              <xsl:with-param name="arc" select="$arc"/>
79              <xsl:with-param name="link" select="$link"/>
80              <xsl:with-param name="via-linkbase" select="$via-linkbase"/>
81          </xsl:apply-templates>
82
83      </xsl:template>
84
85
86  <!---
87      Process XLink resource- and locator-type elements.
88      @param element()   $arc           XLink arc-type element (passed by previous template).
89      @param element()   $link          XLink extended-type elements (the matched element's parent).
90      @param element()*  $linkbase      Set of XLink locator-type elements from a linkbase that might
            reference to the context node.
91      @param xs:boolean  $via-linkbase  Signalizing whether the context node is part of a referenced
            linkbase or not.
92  -->
93      <xsl:template match="*[@xlink:type = 'resource' or @xlink:type = 'locator']" mode="process-link">
94          <xsl:param name="arc"          as="element()"/>
95          <xsl:param name="link"         as="element()"  select=".."/>
96          <xsl:param name="linkbase"     as="element()*" select="()" tunnel="yes"/>
97          <xsl:param name="via-linkbase" as="xs:boolean" select="false()"/>
```

```
98
99          <xsl:param name="starting-resource" select="."/>
100         <xsl:param name="ending-resources" select="$link/*[@xlink:label = $arc/@xlink:to]"/>
101
102         <!-- TODO: differentiate by 'show'/'actuate' attribute and resolve conflict
                [REC-xlink11-20100506]: "Any show attribute value on a linkbase arc must be ignored ..." -->
103         <xsl:if test="$arc/@xlink:show != 'none'">
104
105             <!-- differentiate by 'arcrole' attribute value -->
106             <xsl:choose>
107                 <!-- linkbase reference processing -->
108                 <xsl:when test="$arc/@xlink:arcrole = 'http://www.w3.org/1999/xlink/properties/linkbase'">
109
110                     <xsl:apply-templates select="$starting-resource" mode="starting-resource">
111                         <xsl:with-param name="linkbase" tunnel="yes" select="$linkbase |
                                document($ending-resources/@xlink:href)//*[@xlink:type='locator']
                                [substring-before(resolve-uri(@xlink:href,base-uri(.)),'#') =
                                substring-before(resolve-uri($starting-resource/@xlink:href,
                                base-uri($starting-resource)),'#')]"/>
112                         <!-- TODO: Sollen die Linkbasen Locator an dieser Stelle tatsächlich gefiltert
                                werden? -->
113                     </xsl:apply-templates>
114
115                 </xsl:when>
116                 <!-- 'normal' link processing -->
117                 <xsl:otherwise>
118
119                     <!-- starting-resource -->
120                     <xsl:if test="$arc/@xlink:show = 'embed' and not($via-linkbase)">
121                         <xsl:apply-templates select="$starting-resource" mode="starting-resource"/>
122                     </xsl:if>
123                     <!-- ending-resources -->
124                     <xsl:apply-templates select="$ending-resources" mode="ending-resource"/>
125
126                 </xsl:otherwise>
127             </xsl:choose>
128
129         </xsl:if>
130
131     </xsl:template>
132
133
134 <!--process-resource-->
135
136 <!---
137     Process XLink resources.
138     @param element()   $context       XLink resource- or locator-type element (the matched element).
139     @param element()*  $linkbase      Set of XLink locator-type elements from a linkbase that might
              reference to the context node.
140 -->
141     <xsl:template match="*[@xlink:type = 'resource' or @xlink:type = 'locator']" name="process-resource"
          mode="starting-resource ending-resource">
142         <xsl:param name="context"  select="." as="element()"/>
143         <xsl:param name="linkbase" select="()" as="element()*" tunnel="yes"/>
144
```

```
145         <!-- process resource -->
146         <xsl:choose>
147             <!-- locator-type -->
148             <xsl:when test="$context/@xlink:type = 'locator'">
149                 <xsl:variable name="resource" select="document($context/@xlink:href)"/>
150                 <xsl:choose>
151                     <xsl:when test="$resource/@xlink:type = 'extended' or $resource/@xlink:type =
                            'simple'">
152                         <!-- if: self == extended or simple, then: process self, otherwise: process parent
                                -->
153                         <xsl:apply-templates select="$resource" mode="process-link"/>
154                     </xsl:when>
155                     <xsl:when test="$resource/@xlink:type = 'arc' or $resource/@xlink:type = 'resource' or
                            $resource/@xlink:type = 'locator'">
156                         <xsl:apply-templates select="$resource/.." mode="process-link">
157                             <xsl:with-param name="link-focus" select="$resource"/>
158                         </xsl:apply-templates>
159                     </xsl:when>
160                     <xsl:otherwise>
161                         <xsl:call-template name="embed">
162                             <xsl:with-param name="context" select="$resource"/>
163                         </xsl:call-template>
164                     </xsl:otherwise>
165                 </xsl:choose>
166             </xsl:when>
167             <!-- resource-type -->
168             <xsl:when test="$context/@xlink:type = 'resource'">
169                 <xsl:call-template name="embed">
170                     <xsl:with-param name="context" select="$context"/>
171                 </xsl:call-template>
172             </xsl:when>
173         </xsl:choose>
174 
175         <!-- process associated linkbases -->
176         <xsl:for-each
                select="$linkbase[dag:is_subset($context,document(resolve-uri(@xlink:href,base-uri(.))))]">
177             <xsl:apply-templates select="./.." mode="process-link">
178                 <xsl:with-param name="via-linkbase" select="true()"/>
179                 <xsl:with-param name="link-focus" select="."/>
180             </xsl:apply-templates>
181         </xsl:for-each>
182         <!-- TODO: Hier werden nur Linkbasen-Referenzen aufgelöst, die auf XLink-Resourcen zeigen - was
                ist mit Referenzen, die von "normalen" Knoten ausgehen?
183              Verschieben nach Identity-Copy? -->
184 
185     </xsl:template>
186 
187 
188 <!--embed-->
189 
190 <!---
191     Embed XLink resources.
192     @param element()   $context       XLink resource- or locator-type element (the matched element).
193 -->
194     <xsl:template name="embed">
```

```
195         <xsl:param name="context" select="." as="element()*"/>
196         <xsl:apply-templates select="$context/node()" mode="process-link"/>
197     </xsl:template>
198 
199 
200 <!---
201     Identity template.
202 -->
203     <xsl:template match="@*|node()" mode="process-link">
204         <xsl:copy>
205             <xsl:apply-templates select="@*|node()" mode="process-link"/>
206         </xsl:copy>
207     </xsl:template>
208 
209 
210 <!--normalize-arcs-->
211 
212 <!---
213     Returns all explicit and implicit XLink arcs.
214     @param element()   $link          XLink extended-type element (the arcs' parent).
215     @param element()*  $arcs          Set of XLink arc-type elements.
216 -->
217     <xsl:template name="get-arc-type-elements" as="element()*">
218         <xsl:param name="link" as="element()"  select="."/>
219         <xsl:param name="arcs" as="element()*" select="$link/*[@xlink:type = 'arc']"/>
220 
221         <xsl:call-template name="expand-arc-type-elements">
222             <xsl:with-param name="labels" select="distinct-values($link/*/@xlink:label)"/>
223             <xsl:with-param name="arcs">
224                 <xsl:choose>
225                     <xsl:when test="empty($arcs)">
226                         <xsl:element name="tmp">
227                             <xsl:attribute name="xlink:type" select="'arc'"/>
228                         </xsl:element>
229                     </xsl:when>
230                     <xsl:otherwise>
231                         <xsl:copy-of select="$arcs"/>
232                     </xsl:otherwise>
233                 </xsl:choose>
234             </xsl:with-param>
235         </xsl:call-template>
236     </xsl:template>
237 
238 <!---
239     Returns the expanded XLink arcs.
240     @param xs:string*  $labels        The XLink link's lables.
241     @param element()*  $arc           Set of XLink arc-type elements.
242 -->
243     <xsl:template name="expand-arc-type-elements" as="element()*">
244         <xsl:param name="labels" as="xs:string*"/>
245         <xsl:param name="arcs"/>
246 
247         <xsl:for-each select="$arcs/*">
248             <xsl:call-template name="create-arc-type-element">
249                 <xsl:with-param name="arc-type-element" select="."/>
```

```
250                 <xsl:with-param name="starting-resources-labels">
251                     <xsl:choose>
252                         <xsl:when test="exists(@xlink:from)">
253                             <xsl:value-of select="@xlink:from"/>
254                         </xsl:when>
255                         <xsl:otherwise>
256                             <xsl:value-of select="$labels"/>
257                         </xsl:otherwise>
258                     </xsl:choose>
259                 </xsl:with-param>
260                 <xsl:with-param name="ending-resources-labels">
261                     <xsl:choose>
262                         <xsl:when test="exists(@xlink:to)">
263                             <xsl:value-of select="@xlink:to"/>
264                         </xsl:when>
265                         <xsl:otherwise>
266                             <xsl:value-of select="$labels"/>
267                         </xsl:otherwise>
268                     </xsl:choose>
269                 </xsl:with-param>
270             </xsl:call-template>
271         </xsl:for-each>
272     </xsl:template>
273 
274 <!---
275     Creates arc-type XLink elements.
276     @param element()   $arc-type-element           The original XLink arc-type element.
277     @param xs:string*  $starting-resources-labels  The arc's starting resources' labels.
278     @param xs:string*  $ending-resources-labels    The arc's ending resources' labels.
279 -->
280     <xsl:template name="create-arc-type-element" as="element()*">
281         <xsl:param name="arc-type-element"          as="element()"/>
282         <xsl:param name="starting-resources-labels" as="xs:string*"/>
283         <xsl:param name="ending-resources-labels"   as="xs:string*"/>
284 
285         <xsl:for-each select="$starting-resources-labels">
286             <xsl:variable name="from" select="."/>
287             <xsl:for-each select="$ending-resources-labels">
288                 <xsl:variable name="to" select="."/>
289                 <xsl:for-each select="$arc-type-element">
290                     <xsl:copy>
291                         <xsl:attribute name="xlink:from" select="$from"/>
292                         <xsl:attribute name="xlink:to" select="$to"/>
293                         <xsl:for-each select="@* except (@xlink:from | @xlink:to) | node()">
294                             <xsl:copy/>
295                         </xsl:for-each>
296                     </xsl:copy>
297                 </xsl:for-each>
298             </xsl:for-each>
299         </xsl:for-each>
300     </xsl:template>
301 
302 </xsl:stylesheet>
```

C.1.2. Function Library for Directed Acyclic Graphs

dag-functions.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=============================================================================================================
    Functions to check if two distinct node references point to the same node or to a subset from one
          another
    @see http://my.safaribooksonline.com/book/xml/0596003722/selecting-and-traversing/xsltckbk-chp-4-sect-2
    @see http://www.xml.com/cookbooks/xsltckbk/solution.csp?day=5
    ============================================================================================================-->
<xsl:stylesheet version="2.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
    exclude-result-prefixes="#all">

<!---
    Check if all the nodes in one node-set are the same as all the nodes in another.
    @param node()* $ns1  1st node-set.
    @param node()* $ns2  2nd node-set.
-->
    <xsl:function name="dag:is_same" as="xs:boolean">
        <xsl:param name="ns1" as="node()*"/>
        <xsl:param name="ns2" as="node()*"/>
        <xsl:value-of select="count($ns1|$ns2) = count($ns1) and count($ns1) = count($ns2)"/>
    </xsl:function>

<!---
    Check if one node-set is either equal to or a subset of another node-set.
    @param node()* $ns1  1st node-set.
    @param node()* $ns2  2nd node-set to be tested if it is a subset of the 1st.
-->
    <xsl:function name="dag:is_subset" as="xs:boolean">
        <xsl:param name="ns1" as="node()*"/>
        <xsl:param name="ns2" as="node()*"/>
        <xsl:value-of select="count($ns1|$ns2) = count($ns1)"/>
    </xsl:function>

<!---
    Check if one node-set is a proper subset of (i.e. not equal to) another node-set.
    @param node()* $ns1  1st node-set.
    @param node()* $ns2  2nd node-set to be tested if it is a subset of the 1st.
-->
    <xsl:function name="dag:is_proper_subset" as="xs:boolean">
        <xsl:param name="ns1" as="node()*"/>
        <xsl:param name="ns2" as="node()*"/>
        <xsl:value-of select="count($ns1|$ns2) = count($ns1) and count($ns1) > count($ns2)"/>
    </xsl:function>

</xsl:stylesheet>
```

C.1.3. Standoff to Intermediate Inline Markup Transformation

xlink-stdoff2inl.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=============================================================================================================
    XSLT stylesheet to transform XLink standoff markup to intermediate inline markup
    @author   Frederik R.N. Schlupkothen
    @version  19.06.2015
    ============================================================================================================-->
<xsl:stylesheet version="2.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
    exclude-result-prefixes="#all">

    <xsl:import href="xlink-traverse.xsl"/>

    <xsl:param name="resolve-basic-units" select="'true'"/>


    <xsl:template match="/" name="process-stdoff">

        <dag:dag>
            <!-- process each multistructured document description -->
            <xsl:for-each select="//*[@xlink:type = 'extended']">
                <xsl:variable name="link" select="."/>

                <!-- process each document structure -->
                <xsl:for-each select="*[@xlink:type = 'arc'][@xlink:show != 'none' and @xlink:actuate !=
                      'none']">
                    <!-- TODO: resolve conflict [REC-xlink11-20100506]: "Any show attribute value on a
                          linkbase arc must be ignored ..." -->
                    <xsl:variable name="arc" select="."/>

                    <dag:dag>
                        <xsl:apply-templates select="$link" mode="process-link">
                            <xsl:with-param name="link-focus" select="$arc"/>
                        </xsl:apply-templates>
                    </dag:dag>

                </xsl:for-each>
            </xsl:for-each>
        </dag:dag>

    </xsl:template>


<!--process-link-->

<!---
    Process XLink arc-type elements that interlink alternative resources.
    @param element()   arc          XLink arc-type element (the matched element).
    @param element()   link         XLink extended-type elements (the matched element's parent).
```

```
50      @param element()?  link-focus    The originally referenced element (arc or resource) within the link.
51      @param xs:boolean  via-linkbase  Signalizing whether the context node is part of a referenced linkbase
              or not.
52  -->
53      <xsl:template match="*[@xlink:type = 'arc'][@xlink:arcrole =
              'http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt']" mode="process-link">
54          <xsl:param name="arc"          as="element()"  select="."/>
55          <xsl:param name="link"         as="element()"  select=".."/>
56          <xsl:param name="link-focus"   as="element()?" select="()"/>
57          <xsl:param name="via-linkbase" as="xs:boolean" select="false()"/>
58
59          <xsl:choose>
60              <xsl:when test="$resolve-basic-units = 'true'">
61                  <dag:alt>
62                      <xsl:next-match>
63                          <xsl:with-param name="arc" select="$arc"/>
64                          <xsl:with-param name="link" select="$link"/>
65                          <xsl:with-param name="link-focus" select="
66                              if (empty($link-focus))
67                                  then $link/*[@xlink:type = 'resource' or @xlink:type = 'locator'][1]
68                                  else $link-focus
69                              "/><!-- each alternative only once -->
70                          <xsl:with-param name="via-linkbase" select="$via-linkbase"/>
71                      </xsl:next-match>
72                  </dag:alt>
73              </xsl:when>
74              <xsl:otherwise>
75                  <!-- output content ID (debug mode) -->
76                  <xsl:value-of select="$link/@*[$link intersect id(.)]"/>
77              </xsl:otherwise>
78          </xsl:choose>
79
80      </xsl:template>
81
82
83  <!---
84      Process XLink resource- and locator-type elements that describe element annotations.
85      @param element()   $arc           XLink arc-type element (passed by previous template).
86      @param element()   $link          XLink extended-type elements (the matched element's parent).
87      @param xs:boolean  $via-linkbase  Signalizing whether the context node is part of a referenced
              linkbase or not.
88  -->
89      <xsl:template match="*[@xlink:type = 'resource' or @xlink:type = 'locator'][@xlink:role =
              'http://www.w3.org/2001/XMLSchema#element']" mode="process-link">
90          <xsl:param name="arc"          as="element()"/>
91          <xsl:param name="link"         as="element()"  select=".."/>
92          <xsl:param name="via-linkbase" as="xs:boolean" select="false()"/>
93
94          <xsl:param name="starting-resource" select="."/>
95          <xsl:param name="ending-resources" select="$link/*[@xlink:label = $arc/@xlink:to]"/>
96
97          <xsl:choose>
98              <xsl:when test="
99                  $arc/@xlink:arcrole = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type' and
100                 $ending-resources/@xlink:role = 'http://www.w3.org/2001/XMLSchema#QName'
```

```
101                 ">
102                 <dag:element>
103                     <xsl:attribute name="type" select="$ending-resources/text()"/>
104                 </dag:element>
105             </xsl:when>
106             <xsl:otherwise>
107                 <xsl:next-match>
108                     <xsl:with-param name="arc" select="$arc"/>
109                     <xsl:with-param name="link" select="$link"/>
110                     <xsl:with-param name="via-linkbase" select="$via-linkbase"/>
111                     <xsl:with-param name="starting-resource" select="$starting-resource"/>
112                     <xsl:with-param name="ending-resources" select="$ending-resources"/>
113                 </xsl:next-match>
114             </xsl:otherwise>
115         </xsl:choose>
116
117     </xsl:template>
118
119
120 <!---
121     Process XLink resource- and locator-type elements that describe attribute annotations.
122     @param element()   $arc           XLink arc-type element (passed by previous template).
123     @param element()   $link          XLink extended-type elements (the matched element's parent).
124     @param xs:boolean  $via-linkbase  Signalizing whether the context node is part of a referenced
            linkbase or not.
125 -->
126     <xsl:template match="*[@xlink:type = 'resource' or @xlink:type = 'locator'][@xlink:role =
          'http://www.w3.org/2001/XMLSchema#attribute']" mode="process-link">
127         <xsl:param name="arc"          as="element()"/>
128         <xsl:param name="link"         as="element()"  select=".."/>
129         <xsl:param name="via-linkbase" as="xs:boolean" select="false()"/>
130
131         <xsl:param name="starting-resource" select="."/>
132         <xsl:param name="ending-resources" select="$link/*[@xlink:label = $arc/@xlink:to]"/>
133
134         <xsl:choose>
135             <xsl:when test="
136                 $arc/@xlink:arcrole = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type' and
137                 $ending-resources/@xlink:role = 'http://www.w3.org/2001/XMLSchema#QName'
138                 ">
139                 <dag:attribute>
140                     <xsl:attribute name="id" select="generate-id($starting-resource)"/>
141                     <xsl:attribute name="type" select="$ending-resources/text()"/>
142                 </dag:attribute>
143             </xsl:when>
144             <xsl:when test="$arc/@xlink:arcrole = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#value'">
145                 <dag:attribute>
146                     <xsl:attribute name="id" select="generate-id($starting-resource)"/>
147                     <xsl:attribute name="value" select="$ending-resources/text()"/>
148                 </dag:attribute>
149             </xsl:when>
150             <xsl:otherwise>
151                 <xsl:next-match>
152                     <xsl:with-param name="arc" select="$arc"/>
153                     <xsl:with-param name="link" select="$link"/>
```

```
154                     <xsl:with-param name="via-linkbase" select="$via-linkbase"/>
155                     <xsl:with-param name="starting-resource" select="$starting-resource"/>
156                     <xsl:with-param name="ending-resources" select="$ending-resources"/>
157                 </xsl:next-match>
158             </xsl:otherwise>
159         </xsl:choose>
160 
161     </xsl:template>
162 
163 
164 <!--process-resource-->
165 
166 <!---
167     Process XLink resource- and locator-type elements that describe structure nodes (default).
168     @param element()   $context       XLink resource- or locator-type element (the matched element).
169 -->
170     <xsl:template match="*[@xlink:type = 'resource' or @xlink:type = 'locator']" mode="ending-resource">
171         <xsl:param name="context" select="." as="element()"/>
172 
173         <dag:node>
174             <xsl:if test="@xlink:role">
175                 <xsl:attribute name="role" select="@xlink:role"/>
176             </xsl:if>
177             <xsl:next-match>
178                 <xsl:with-param name="context" select="$context"/>
179             </xsl:next-match>
180         </dag:node>
181 
182     </xsl:template>
183 
184 </xsl:stylesheet>
```

C.1.4. Intermediate to Final Inline Markup Transformation

`xlink-inl2final.xsl`

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--=============================================================================================================
3      XSLT stylesheet to copy documents and transform embedded intermediate inline markup to final markup
4      @author   Frederik R.N. Schlupkothen
5      @version  19.06.2015
6      ============================================================================================================-->
7  <xsl:stylesheet version="2.0"
8      xmlns:xs="http://www.w3.org/2001/XMLSchema"
9      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
10     xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
11     exclude-result-prefixes="#all">
12 
13     <xsl:param name="resolve-content-mode" select="'true'"/>
14 
15 
16     <xsl:template match="/" name="process-inline">
17         <xsl:apply-templates mode="process-dag"/>
```

```
18      </xsl:template>
19
20
21  <!---
22      Identity template for non-intermediate annotations.
23  -->
24      <xsl:template match="node()[not(self::dag:*)]|@*" mode="process-dag">
25          <xsl:copy>
26              <xsl:apply-templates select="node()|@*" mode="process-dag"/>
27          </xsl:copy>
28      </xsl:template>
29
30
31  <!---
32      Transform element node descriptions to XML elements.
33  -->
34      <xsl:template match="dag:*[dag:element]" mode="process-dag">
35          <xsl:choose>
36              <xsl:when test="count(dag:element) &gt; 1">
37                  <xsl:message>Warning (ambiguous element type): skipping annotation (<xsl:value-of
                        select="dag:element/@type"/>) - there is more than one element type assigned to the
                        current document node</xsl:message>
38                  <xsl:apply-templates mode="process-dag"/>
39              </xsl:when>
40              <xsl:otherwise>
41                  <xsl:element name="{dag:element/@type}" namespace="{dag:resolve-namespace(dag:element)}">
42                      <xsl:apply-templates select="dag:attribute[not(starts-with(@type,'xmlns'))]"
                            mode="process-dag-attr"/>
43                      <xsl:apply-templates mode="process-dag"/>
44                  </xsl:element>
45              </xsl:otherwise>
46          </xsl:choose>
47      </xsl:template>
48
49
50  <!---
51      Transform attribute node descriptions to XML attributes.
52  -->
53      <xsl:template match="dag:attribute[@type]" mode="process-dag-attr">
54          <xsl:variable name="elem-namespace" select="dag:resolve-namespace(../dag:element[1])"/>
55          <xsl:variable name="attr-namespace" select="dag:resolve-namespace(.)"/>
56          <xsl:choose>
57              <xsl:when test="starts-with( lower-case( normalize-space( @type ) ), 'xml:' )">
58                  <xsl:attribute name="{@type}" select="../dag:attribute[@id = current()/@id]/@value"/>
59              </xsl:when>
60              <xsl:when test="$attr-namespace eq $elem-namespace">
61                  <xsl:attribute name="{if (contains(@type,':')) then substring-after(@type,':') else
                        @type}" select="../dag:attribute[@id = current()/@id]/@value"/>
62              </xsl:when>
63              <xsl:otherwise>
64                  <xsl:attribute name="{@type}" namespace="{dag:resolve-namespace(.)}"
                        select="../dag:attribute[@id = current()/@id]/@value"/>
65              </xsl:otherwise>
66          </xsl:choose>
67      </xsl:template>
```

```
68
69
70  <!---
71      Resolve namespace for XML elements and attributes.
72      @param element()  context  Given intermediate markup description element.
73  -->
74      <xsl:function name="dag:resolve-namespace">
75          <xsl:param name="context" as="element()"/>
76          <xsl:variable name="namespace-prefix" select="substring-before($context/@type,':')"/>
77          <xsl:choose>
78              <xsl:when test="$namespace-prefix = ''">
79                  <xsl:value-of select="root($context)//dag:attribute[@value][@id =
                        $context/ancestor::dag:*/dag:attribute[@type = 'xmlns'][1]/@id]/@value"/>
80              </xsl:when>
81              <xsl:otherwise>
82                  <xsl:value-of select="root($context)//dag:attribute[@value][@id =
                        $context/ancestor::dag:*/dag:attribute[@type =
                        concat('xmlns:',$namespace-prefix)][1]/@id]/@value"/>
83              </xsl:otherwise>
84          </xsl:choose>
85      </xsl:function>
86
87
88  <!---
89      Transform alternative content node descriptions by choosing one content node by content mode or
            pass-through the intermediate markup (depending on $resolve-content-mode).
90      @param xs:string|null  content-mode  Given content mode identifier (e.g. URI).
91  -->
92      <xsl:template match="dag:alt" mode="process-dag">
93          <xsl:param name="content-mode" tunnel="yes"/>
94          <xsl:choose>
95              <xsl:when test="$resolve-content-mode = 'true' or $content-mode">
96                  <xsl:variable name="mode" select="if ($content-mode) then $content-mode else ../@role"/>
97                  <xsl:choose>
98                      <xsl:when test="not($mode)">
99                          <xsl:message>Warning (no assigned content mode): random content mode output
                                (<xsl:value-of select="*[1]/@role"/>) - there is no assigned content mode;
                                outputting first defined content representation instead</xsl:message>
100                         <xsl:apply-templates select="*[1]" mode="process-dag"/>
101                     </xsl:when>
102                     <xsl:when test="count(*[@role = $mode]) = 0">
103                         <xsl:message>Warning (no matching content mode): random content mode output
                                (<xsl:value-of select="*[1]/@role"/>) - there is no matching content
                                representation (<xsl:value-of select="*/@role"/>) for the assigned content
                                mode (<xsl:value-of select="$mode"/>); outputting first defined content
                                representation instead</xsl:message>
104                         <xsl:apply-templates select="*[1]" mode="process-dag"/>
105                     </xsl:when>
106                     <xsl:when test="count(*[@role = $mode]) &gt; 1">
107                         <xsl:message>Warning (matching multiple content nodes): multiple output
                                (<xsl:value-of select="count(*[@role = $mode])"/>) - there are multiple
                                matching content representations for the assigned content mode
                                (<xsl:value-of select="$mode"/>)</xsl:message>
108                         <xsl:apply-templates select="*[@role = $mode]" mode="process-dag"/>
109                     </xsl:when>
```

```
                    <xsl:otherwise>
                        <xsl:apply-templates select="*[@role = $mode]" mode="process-dag"/>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
                <xsl:copy-of select="self::dag:alt"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>

</xsl:stylesheet>
```

C.1.5. Namespace Reorganization

xml-cleanup.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=============================================================================================================
    XSLT stylesheet to copy XML documents and reorganizing (minimizing) namespace-declarations
    @author   Frederik R.N. Schlupkothen
    @version  10.07.2015
    ============================================================================================================-->
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    exclude-result-prefixes="#all">

    <xsl:template match="/" name="xml-cleanup">
        <xsl:apply-templates mode="finalize"/>
    </xsl:template>

    <!-- identity copy of all nodes except elements and attributes -->
    <xsl:template match="node()[not(self::*)]" mode="finalize">
        <xsl:copy>
            <xsl:apply-templates select="node()" mode="finalize"/>
        </xsl:copy>
    </xsl:template>

    <!-- "copy" elements but reorganize namespace declarations -->
    <xsl:template match="*" mode="finalize">
        <xsl:element name="{local-name()}" namespace="{namespace-uri()}">
            <!--xsl:apply-templates select="@*[name() != 'xml:space'],node()" mode="finalize"/-->
            <xsl:apply-templates select="@*,node()" mode="finalize"/>
        </xsl:element>
    </xsl:template>

    <!-- "copy" attributes but reorganize namespace declarations -->
    <xsl:template match="@*" mode="finalize">
        <xsl:attribute name="{local-name()}" namespace="{if (namespace-uri() !=
              namespace-uri(parent::node())) then namespace-uri() else ()}" select="."/>
    </xsl:template>

</xsl:stylesheet>
```

C.1.6. Standoff to Inline Markup Pipeline

xlink-stdoff2final.xpl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--============================================================================================================
    XProc pipeline to transform XLink-based standoff markup to its inline representation
    @author   Frederik R.N. Schlupkothen
    @version  10.07.2015
    ============================================================================================================-->
<p:declare-step version="1.0"
    xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:c="http://www.w3.org/ns/xproc-step"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
    type="dag:stdoff2inl">

    <p:input port="source"/>
    <p:output port="result"/>

    <!-- transform standoff to intermediate inline markup -->
    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="xlink-stdoff2inl.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
    </p:xslt>

    <!-- transform intermediate to final inline markup -->
    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="xlink-inl2final.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
    </p:xslt>

    <!-- reorganize namespace declarations -->
    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="xml-cleanup.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
    </p:xslt>

</p:declare-step>
```

C.2. Rhetorical Structure Theory Processing

C.2.1. Flat RST Document Type Definition

`RST.dtd` (O'Donnell 2000)

```
<!ELEMENT rst (header, body)>
<!ELEMENT header (relations)>
<!ELEMENT relations (rel)*>
<!ATTLIST relations
          file CDATA #IMPLIED
>
<!ELEMENT body (segment | group)*>
<!ELEMENT rel EMPTY>
<!ATTLIST rel
          name CDATA #REQUIRED
          type (rst | multinuc) #REQUIRED
>
<!ELEMENT segment (#PCDATA)>
<!ATTLIST segment
          id ID #IMPLIED
          parent IDREF #REQUIRED
          relname CDATA #REQUIRED
>
<!ELEMENT group EMPTY>
<!ATTLIST group
          id ID #IMPLIED
          type (multinuc | span | constit) #REQUIRED
          parent IDREF #IMPLIED
          relname CDATA #IMPLIED
>
```

C.2.2. Deep RST Document Type Definition

`deepRST.dtd`

```
<!ELEMENT group   (group|segment)*>
<!ELEMENT segment (#PCDATA|group|segment)*>
<!ENTITY % attr
         "xmlns    CDATA #FIXED 'http://www.sfu.ca/rst'
          relname  CDATA #IMPLIED
          reltype (rst|multinuc) #IMPLIED" >
<!ATTLIST group   %attr;>
<!ATTLIST segment %attr;>
```

C.2.3. Deep RST Schematron Schema

rst.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=============================================================================================================
    Schematron schema to report and validate deepRST.dtd compliant RST structures
    @author   Frederik R.N. Schlupkothen
    @version  01.12.2015
    =============================================================================================================-->
<schema queryBinding="xslt2"
    xmlns="http://purl.oclc.org/dsdl/schematron"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:include href="rst-functions.xsl"/>

    <ns uri="http://dmt.uni-wuppertal.de/schlupko/xlink4dags" prefix="dag"/>
    <ns uri="http://www.sfu.ca/rst" prefix="rst"/>

    <pattern id="gen">
        <title>General RST Reporting and Validation</title>
        <rule context="/">
            <!-- root reporting -->
            <report role="info" test=".">The given structure contains <value-of select="count(
                  //rst:segment )"/> text node(s)</report>
            <!-- root validation -->
            <assert role="error" test="rst:group">The root element must be of type "rst:group"</assert>
        </rule>
        <rule context="rst:*">
            <!-- structure reporting -->
            <report role="info" test=".">&#x2022; <name/> &#x2192; element(/<value-of select="string-join(
                  ancestor-or-self::rst:*/string( count( preceding-sibling::rst:* ) + 1 ), '/'
                  )"/>)</report>
            <report role="info" test="rst:is_nucleus(.)"><name/> is a nucleus</report>
            <report role="info" test="rst:is_satellite(.)"><name/> is a satellite</report>
            <report role="info" test="rst:is_multinuclear(.)"><name/> is multinuclear</report>
            <report role="info" test="@relname and rst:is_satellite(.)"><name/> is in "<value-of
                  select="@relname"/>" relation to its nucleus</report>
            <report role="info" test="@relname and rst:is_multinuclear(.)"><name/> is in "<value-of
                  select="@relname"/>" relation to its co-nuclei</report>
            <report role="info" test="self::rst:segment"><name/>'s text is "<value-of
                  select="normalize-space( string-join( self::*/text(), '' ) )"/>"</report>
            <report role="info" test="count( rst:* )"><name/> has <value-of select="count( rst:* )"/>
                  child(ren)</report>
            <!-- structure validation -->
            <assert role="error" test="( @relname and @reltype ) or not( @relname or @reltype )">If there
                  is a "@relname" attribute there has to be a "@reltype" attribute and vice versa</assert>
            <report role="error" test="self::rst:group and rst:contains_text(.)"><name/> must not have
                  text content</report>
            <report role="error" test="self::rst:segment and not( rst:contains_text(.) )"><name/> must
                  have text content</report>
            <assert role="error" test="count( rst:get_text(.) | rst:*[rst:is_nucleus(.)] |
                  rst:get_multinuclear_group(.) ) = 1">The sum of text nodes, (non-multinuclear) nuclei,
                  and multineclear related children groups in a span is 1</assert>
```

```
            <assert role="error" test="count( distinct-values(
                  rst:get_multinuclear_group(.)/rst:*/@relname ) ) &lt;= 1">All multinuclear related
                  children within a group must have the same "@relname" value</assert>
        </rule>
    </pattern>

</schema>
```

C.2.4. Function Library for RST Trees

rst-functions.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=============================================================================================================
    XSLT function library for RST specific structure processing of deepRST.dtd compliant documents
    @author   Frederik R.N. Schlupkothen
    @version  01.12.2015
    ============================================================================================================-->
<xsl:stylesheet version="2.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:rst="http://www.sfu.ca/rst"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
    exclude-result-prefixes="#all">

    <xsl:include href="dag-functions.xsl"/>


<!--text-->

<!---
    Test if a given node contains non-trivial text nodes.
    @param node()  node  Node to be tested.
-->
    <xsl:function name="rst:contains_text" as="xs:boolean">
        <xsl:param name="node" as="node()"/>
        <xsl:value-of select="count(rst:get_text($node)) != 0"/>
    </xsl:function>


<!---
    Select non-trivial text nodes.
    @param node()  node  Context node for potential text nodes.
-->
    <xsl:function name="rst:get_text" as="node()*">
        <xsl:param name="node" as="node()"/>
        <xsl:sequence select="$node/text()[normalize-space(string-join(.,'')) != '']"/>
    </xsl:function>


<!--relation (general)-->

```

```
<!---
    Test if a given node is in a given RST relation.
    @param node()     node  Node to be tested.
    @param xs:string  name  Name of the RST relation.
-->
    <xsl:function name="rst:has_relation" as="xs:boolean">
        <xsl:param name="node" as="node()"/>
        <xsl:param name="name" as="xs:string"/>
        <xsl:value-of select="$node/self::rst:*/upper-case(@relname) = upper-case($name)"/>
    </xsl:function>


<!--nuclei-->

<!---
    Test if a given node is a RST nucleus (non-multinuclear).
    @param node()  node  Node to be tested.
-->
    <xsl:function name="rst:is_nucleus" as="xs:boolean">
        <xsl:param name="node" as="node()"/>
        <xsl:value-of select="$node/self::rst:*/not(@reltype)"/>
    </xsl:function>


<!---
    Test if two nodes are connected by a chain of RST nuclei (non-multinuclear).
    @param node()  first  Source node of the chain.
    @param node()  last   Sink node of the chain.
-->
    <xsl:function name="rst:is_nucleus_chain" as="xs:boolean">
        <xsl:param name="first" as="node()"/>
        <xsl:param name="last"  as="node()"/>
        <xsl:variable name="chain" select="rst:get_chain( $first, $last )" as="node()*"/>
        <xsl:value-of select="dag:is_same( $chain, $chain[rst:is_nucleus(.)] )"/>
    </xsl:function>


<!---
    Select all subsequently chained RST nuclei (non-multinuclear).
    @param node()  node  Source node of the potential nucleus chain.
-->
    <xsl:function name="rst:traverse_nucleus_chain" as="node()*">
        <xsl:param name="node" as="node()*"/>
        <xsl:for-each select="$node">
            <xsl:choose>
                <xsl:when test="rst:*[rst:is_nucleus(.)]">
                    <xsl:sequence select="rst:traverse_nucleus_chain(rst:*[rst:is_nucleus(.)])"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:sequence select="."/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:for-each>
    </xsl:function>

```

```
96
97  <!---
98      Select all subsequently chained nodes.
99      @param node()   first  Source node of the chain.
100     @param node()*  last   Sink node of the chain.
101 -->
102     <xsl:function name="rst:get_chain" as="node()*">
103         <xsl:param name="first" as="node()"/>
104         <xsl:param name="last"  as="node()*"/>
105         <xsl:sequence select="($first/descendant-or-self::rst:* intersect $last/ancestor-or-self::rst:*)"/>
106     </xsl:function>
107
108
109 <!--satellites-->
110
111 <!---
112     Test if a given node is a RST satellite.
113     @param node()  node  Node to be tested.
114 -->
115     <xsl:function name="rst:is_satellite" as="xs:boolean">
116         <xsl:param name="node" as="node()"/>
117         <xsl:value-of select="$node/self::rst:*/@reltype = 'rst'"/>
118     </xsl:function>
119
120
121 <!---
122     Test if a given node is a RST satellite in a given relation to its nucleus.
123     @param node()     node  Node to be tested.
124     @param xs:string  name  Name of the RST relation.
125 -->
126     <xsl:function name="rst:is_satellite" as="xs:boolean">
127         <xsl:param name="node" as="node()"/>
128         <xsl:param name="name" as="xs:string"/>
129         <xsl:value-of select="rst:is_satellite($node) and rst:has_relation($node,$name)"/>
130     </xsl:function>
131
132
133 <!---
134     Test if a given node has a RST satellite.
135     @param node()  node  Node to be tested.
136 -->
137     <xsl:function name="rst:has_satellite" as="xs:boolean">
138         <xsl:param name="node" as="node()"/>
139         <xsl:value-of select="exists($node/rst:*[rst:is_satellite(.)])"/>
140     </xsl:function>
141
142
143 <!---
144     Test if a given node has a RST satellite in a given relation.
145     @param node()     node  Node to be tested.
146     @param xs:string  name  Name of the RST relation.
147 -->
148     <xsl:function name="rst:has_satellite" as="xs:boolean">
149         <xsl:param name="node" as="node()"/>
150         <xsl:param name="name" as="xs:string"/>
```

```
151         <xsl:value-of select="exists($node/rst:*[rst:is_satellite(.,$name)])"/>
152     </xsl:function>
153
154
155 <!---
156     Test if a given node has a preceding RST satellite.
157     @param node()  node  Node to be tested.
158 -->
159     <xsl:function name="rst:has_preceding_satellite" as="xs:boolean">
160         <xsl:param name="node" as="node()"/>
161         <xsl:value-of select="$node/count( ( rst:*[not(rst:is_satellite(.))] | rst:get_text(.)
                )[1]/preceding-sibling::rst:* ) &gt; 0"/>
162     </xsl:function>
163
164
165 <!---
166     Test if a given node has a preceding RST satellite in a given relation.
167     @param node()     node  Node to be tested.
168     @param xs:string  name  Name of the RST relation.
169 -->
170     <xsl:function name="rst:has_preceding_satellite" as="xs:boolean">
171         <xsl:param name="node" as="node()"/>
172         <xsl:param name="name" as="xs:string"/>
173         <xsl:value-of select="$node/count( ( rst:*[not(rst:is_satellite(.))] | rst:get_text(.)
                )[1]/preceding-sibling::rst:*[rst:is_satellite(.,$name)] ) &gt; 0"/>
174     </xsl:function>
175
176
177 <!---
178     Select all preceding RST satellites.
179     @param node()*  nodes  Context nodes of potential RST satellites.
180 -->
181     <xsl:function name="rst:get_preceding_satellite" as="node()*">
182         <xsl:param name="nodes" as="node()*"/>
183         <xsl:sequence select="$nodes/( rst:*[not(rst:is_satellite(.))] | rst:get_text(.)
                )[1]/preceding-sibling::rst:*"/>
184     </xsl:function>
185
186
187 <!---
188     Test if a given node has a following RST satellite.
189     @param node()  node  Node to be tested.
190 -->
191     <xsl:function name="rst:has_following_satellite" as="xs:boolean">
192         <xsl:param name="node" as="node()"/>
193         <xsl:value-of select="$node/count( ( rst:*[not(rst:is_satellite(.))] | rst:get_text(.)
                )[last()]/following-sibling::rst:* ) &gt; 0"/>
194     </xsl:function>
195
196
197 <!---
198     Test if a given node has a following RST satellite in a given relation.
199     @param node()     node  Node to be tested.
200     @param xs:string  name  Name of the RST relation.
201 -->
```

```
202     <xsl:function name="rst:has_following_satellite" as="xs:boolean">
203         <xsl:param name="node" as="node()"/>
204         <xsl:param name="name" as="xs:string"/>
205         <xsl:value-of select="$node/count( ( rst:*[not(rst:is_satellite(.))] | rst:get_text(.)
                )[last()]/following-sibling::rst:*[rst:is_satellite(.,$name)] ) &gt; 0"/>
206     </xsl:function>
207
208
209 <!---
210     Select all following RST satellites.
211     @param node()*  nodes  Context nodes of potential RST satellites.
212 -->
213     <xsl:function name="rst:get_following_satellite" as="node()*">
214         <xsl:param name="nodes" as="node()*"/>
215         <xsl:sequence select="$nodes/( rst:*[not(rst:is_satellite(.))] | rst:get_text(.)
                )[last()]/following-sibling::rst:*"/>
216     </xsl:function>
217
218
219 <!--multinuclear-->
220
221 <!---
222     Test if a given node is in multinuclear RST relation to its co-nuclei.
223     @param node()  node  Node to be tested.
224 -->
225     <xsl:function name="rst:is_multinuclear" as="xs:boolean">
226         <xsl:param name="node" as="node()"/>
227         <xsl:value-of select="$node/self::rst:*/@reltype = 'multinuc'"/>
228     </xsl:function>
229
230
231 <!---
232     Test if a given node is in a given multinuclear RST relation to its co-nuclei.
233     @param node()     node  Node to be tested.
234     @param xs:string  name  Name of the RST relation.
235 -->
236     <xsl:function name="rst:is_multinuclear" as="xs:boolean">
237         <xsl:param name="node" as="node()"/>
238         <xsl:param name="name" as="xs:string"/>
239         <xsl:value-of select="rst:is_multinuclear($node) and rst:has_relation($node,$name)"/>
240     </xsl:function>
241
242
243 <!---
244     Select all multinuclear related RST nuclei in order of their according group (Note: Sum of groups
          should be 1).
245     @param node()  node  Grouping node of potentially multinuclear related RST nuclei
246 -->
247     <xsl:function name="rst:get_multinuclear_group" as="node()*">
248         <xsl:param name="node" as="node()"/>
249         <xsl:for-each-group select="$node/rst:*" group-adjacent="boolean(rst:is_multinuclear(.))">
250             <xsl:if test="current-grouping-key()">
251                 <dag:group>
252                     <xsl:sequence select="current-group()"/>
253                 </dag:group>
```

```
254             </xsl:if>
255         </xsl:for-each-group>
256     </xsl:function>
257 
258 </xsl:stylesheet>
```

C.2.5. Flat to Deep RST Transformation

flat2deepRST.xsl

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--==========================================================================================================
3      XSLT stylesheet to transform the "flat", original RSTTool format to a "deep", nested equivalent
4      @author   Frederik R.N. Schlupkothen
5      @version  12.11.2014
6      ==========================================================================================================-->
7  <xsl:stylesheet version="2.0"
8      xmlns:xs="http://www.w3.org/2001/XMLSchema"
9      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
10     xmlns="http://www.sfu.ca/rst"
11     exclude-result-prefixes="#all">
12 
13     <xsl:output method="xml" doctype-system="deepRST.dtd" indent="no"/>
14 
15     <xsl:template match="/">
16         <xsl:apply-templates select="//body/*[not(@parent)]"/>
17     </xsl:template>
18 
19     <xsl:template match="group | segment">
20         <xsl:element name="{name()}">
21             <xsl:if test="@relname and @relname != 'span'">
22                 <xsl:attribute name="relname" select="@relname"/>
23                 <xsl:attribute name="reltype" select="root()//header//*[@name =
                       current()/@relname]/@type"/>
24             </xsl:if>
25             <xsl:apply-templates select="root()//body/*[@parent = current()/@id] | text()"/>
26         </xsl:element>
27     </xsl:template>
28 
29 </xsl:stylesheet>
```

C.2.6. Deep to Flat RST Transformation

deep2flatRST.xsl

```
<?xml version="1.0" encoding="utf-8"?>
<!--=============================================================================================================
    XSLT stylesheet to transform the "deep", nested RST representation to the "flat", original RSTTool
          format
    @author   Frederik R.N. Schlupkothen
    @version  29.06.2015
    ============================================================================================================-->
<xsl:stylesheet version="2.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:rst="http://www.sfu.ca/rst"
    exclude-result-prefixes="#all">

    <!-- xsl:output method="xml" doctype-system="RST.dtd" indent="yes"/ --><!-- the RSTTool does not cope
          with DOCTYPE declarations -->
    <xsl:output method="xml" omit-xml-declaration="yes" indent="yes"/>

    <!-- create document's main structure -->
    <xsl:template match="/">
        <rst>
            <header>
                <relations>
                    <xsl:apply-templates select="//rst:*[@relname][not(@relname =
                          preceding::rst:*/@relname or @relname = ancestor::rst:*/@relname)]"
                          mode="header"/>
                </relations>
            </header>
            <body>
                <xsl:apply-templates/>
            </body>
        </rst>
    </xsl:template>

    <!-- relation definitions -->
    <xsl:template match="rst:*" mode="header">
        <rel name="{@relname}" type="{@reltype}"/>
    </xsl:template>

    <!-- grouping spans -->
    <xsl:template match="rst:group">
        <xsl:call-template name="create-rst-element">
            <xsl:with-param name="context-element" select="."/>
        </xsl:call-template>
        <xsl:apply-templates/>
    </xsl:template>

    <!-- text spans -->
    <xsl:template match="text()[normalize-space() != ''][parent::rst:segment]">
        <xsl:call-template name="create-rst-element">
            <xsl:with-param name="context-element" select=".."/>
        </xsl:call-template>
```

```
48      </xsl:template>
49
50      <!-- suppress default text output -->
51      <xsl:template match="text()"/>
52
53      <!-- create span description -->
54      <xsl:template name="create-rst-element">
55          <xsl:param name="context-element" as="element()"/>
56          <xsl:element name="{name($context-element)}">
57              <xsl:attribute name="id" select="rst:get-numeric-id($context-element)"/>
58              <xsl:if test="$context-element/self::rst:group">
59                  <xsl:attribute name="type" select="if ($context-element/rst:*/@reltype = 'multinuc') then
                        'multinuc' else 'span'"></xsl:attribute>
60              </xsl:if>
61              <xsl:if test="$context-element/parent::rst:*">
62                  <xsl:attribute name="parent" select="rst:get-numeric-id($context-element/parent::*)"/>
63                  <xsl:attribute name="relname" select="if ($context-element/@relname) then
                        $context-element/@relname else 'span'"/>
64              </xsl:if>
65              <xsl:if test="$context-element/self::rst:segment">
66                  <xsl:value-of select="normalize-space($context-element/text())"/>
67              </xsl:if>
68          </xsl:element>
69      </xsl:template>
70
71      <!-- create unique number ID -->
72      <xsl:function name="rst:get-numeric-id" as="xs:integer">
73          <xsl:param name="node" as="node()"/>
74          <xsl:value-of select="count($node/preceding::rst:*) + count($node/ancestor::rst:*)"/>
75      </xsl:function>
76
77  </xsl:stylesheet>
```

C.2.7. Deep RST to SVG Transformation

deepRST2SVG.xsl

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!--=========================================================================================================
3       XSLT stylesheet to transform a "deep", nested RST description to a visual SVG representation
4       @author   Frederik R.N. Schlupkothen
5       @version  29.06.2015
6       =========================================================================================================-->
7   <xsl:stylesheet version="2.0"
8       xmlns:xs="http://www.w3.org/2001/XMLSchema"
9       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
10      xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
11      xmlns:rst="http://www.sfu.ca/rst"
12      xmlns="http://www.w3.org/2000/svg"
13      exclude-result-prefixes="#all">
```

```
14
15      <xsl:output
16          method="xml" version="1.0" encoding="UTF-8" indent="yes"
17          doctype-public="-//W3C//DTD SVG 1.1//EN"
18          doctype-system="http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"/>
19
20      <!-- output parameters -->
21      <xsl:param name="span-height-mm"  select="12"     as="xs:double"/><!-- in mm -->
22      <xsl:param name="span-width-mm"   select="9"      as="xs:double"/><!-- in mm -->
23      <xsl:param name="span-margin-mm"  select="2"      as="xs:double"/><!-- in mm -->
24      <xsl:param name="graph-margin-mm" select="2"      as="xs:double"/><!-- in mm -->
25      <xsl:param name="stroke-width-pt" select=".5"     as="xs:double"/><!-- in pt -->
26      <xsl:param name="label-size-pt"   select="10"     as="xs:double"/><!-- in pt -->
27      <xsl:param name="text-size-pt"    select="8"      as="xs:double"/><!-- in pt -->
28      <xsl:param name="dpi"             select="300"    as="xs:integer"/><!-- dots per inch -->
29      <xsl:param name="show-text"       select="true()" as="xs:boolean"/><!-- text or numeric IDs -->
30
31      <!-- convert from mm to px union -->
32      <xsl:function name="rst:mm-to-px" as="xs:integer">
33          <xsl:param name="no-in-mm" as="xs:double"/>
34          <xsl:value-of select="round($no-in-mm * .04 * $dpi)"/>
35      </xsl:function>
36
37      <!-- output sizes -->
38      <xsl:variable name="span-height"  select="rst:mm-to-px($span-height-mm)" as="xs:integer"/>
39      <xsl:variable name="span-width"   select="rst:mm-to-px($span-width-mm)" as="xs:integer"/>
40      <xsl:variable name="span-margin"  select="rst:mm-to-px($span-margin-mm)" as="xs:integer"/>
41      <xsl:variable name="graph-margin" select="rst:mm-to-px($graph-margin-mm)" as="xs:integer"/>
42      <xsl:variable name="label-size"   select="rst:mm-to-px(.3527 * $label-size-pt)" as="xs:integer"/>
43      <xsl:variable name="text-size"    select="rst:mm-to-px(.3527 * $text-size-pt)" as="xs:integer"/>
44
45      <!-- create document's main structure -->
46      <xsl:template match="/">
47          <xsl:variable name="graph-width-mm"  select="count(//rst:segment) * ($span-width-mm +
                $span-margin-mm) - $span-margin-mm + 2 * $graph-margin-mm"/>
48          <xsl:variable name="graph-height-mm" select="max(//rst:segment/rst:get-node-level(self::*)) *
                $span-height-mm + .3527 * $text-size-pt + 2 * $graph-margin-mm"/>
49          <svg version="1.1" stroke="black" font-size="{$label-size}" width="{$graph-width-mm}mm"
                height="{$graph-height-mm}mm"  viewBox="0 0 {rst:mm-to-px($graph-width-mm)}
                {rst:mm-to-px($graph-height-mm)}">
50              <defs>
51                  <marker orient="auto" refY="{.25 * $label-size}" refX="{.4 * $label-size}"
                        markerHeight="{.5 * $label-size}" markerWidth="{.5 * $label-size}"
                        markerUnits="userSpaceOnUse" id="arrowhead-graph">
52                      <path fill="black" d="M {.1 * $label-size} {.1 * $label-size} L {.4 * $label-size}
                            {.25 * $label-size} L {.1 * $label-size} {.4 * $label-size} z"/>
53                  </marker>
54              </defs>
55              <xsl:apply-templates/>
56          </svg>
57      </xsl:template>
58
59      <!-- transform spans -->
60      <xsl:template match="rst:*">
61          <xsl:call-template name="draw-node"/>
```

```
62          <xsl:apply-templates/>
63      </xsl:template>
64
65      <!-- create visual span and arc representations -->
66      <xsl:template name="draw-node">
67          <xsl:param name="x" select="rst:get-x-pos(.)" as="xs:integer"/>
68          <xsl:param name="y" select="rst:get-y-pos(.)" as="xs:integer"/>
69          <xsl:param name="w" select="rst:get-width(.)" as="xs:integer"/>
70
71          <!-- draw node -->
72          <line x1="{$x}" y1="{$y}" x2="{$x + $w}" y2="{$y}" stroke-width="{rst:mm-to-px(.3527 *
                $stroke-width-pt)}"/>
73          <xsl:if test="self::rst:segment">
74              <text x="{$x + .5 * $w}" y="{$y}" dy="{$text-size}" text-anchor="middle"
                    font-size="{$text-size}">
75                  <xsl:choose>
76                      <xsl:when test="$show-text">
77                          <xsl:value-of select="text()"/>
78                      </xsl:when>
79                      <xsl:otherwise>
80                          <xsl:value-of select="rst:get-node-no(.) + 1"/>
81                      </xsl:otherwise>
82                  </xsl:choose>
83              </text>
84          </xsl:if>
85
86          <!-- draw arcs -->
87          <xsl:if test="parent::*">
88              <xsl:variable name="text-num-start" select="rst:get-node-no(parent::*) + 1"/>
89              <xsl:variable name="text-num-end"   select="rst:get-node-range(parent::*) - 1"/>
90              <xsl:variable name="arc-x-pos"   select="rst:get-arc-x-pos(.)"/>
91              <xsl:variable name="arc-x-pos-p" select="rst:get-arc-x-pos(parent::*)"/>
92
93              <xsl:choose>
94                  <xsl:when test="rst:is_nucleus(.)">
95                      <line x1="{$arc-x-pos}" y1="{$y - $span-height + 1.2 * $label-size}" x2="{$arc-x-pos}"
                            y2="{$y}" stroke-width="{rst:mm-to-px(.3527 * $stroke-width-pt)}"/>
96                      <text x="{$arc-x-pos}" y="{$y - $span-height}" dy="{$label-size}" text-anchor="middle">
97                          <xsl:value-of select="$text-num-start"/>
98                          <xsl:if test="$text-num-end &gt; 0">
99                              <xsl:text>&#x2013;</xsl:text>
100                             <xsl:value-of select="$text-num-start + $text-num-end"/>
101                         </xsl:if>
102                     </text>
103                 </xsl:when>
104                 <xsl:when test="rst:is_multinuclear(.)">
105                     <line x1="{$arc-x-pos-p}" y1="{$y - $span-height + 1.2 * $label-size}"
                            x2="{$arc-x-pos}" y2="{$y}" stroke-width="{rst:mm-to-px(.3527 *
                            $stroke-width-pt)}"/>
106                     <xsl:if test="not(preceding-sibling::rst:*[1][rst:is_multinuclear(.)])">
107                         <text x="{$arc-x-pos-p}" y="{$y - $span-height}" dy="{$label-size}"
                                text-anchor="middle">
108                             <xsl:value-of select="$text-num-start"/>
109                             <xsl:if test="$text-num-end &gt; 0">
110                                 <xsl:text>&#x2013;</xsl:text>
```

```
111                             <xsl:value-of select="$text-num-start + $text-num-end"/>
112                         </xsl:if>
113                     </text>
114                     <text x="{$arc-x-pos-p}" y="{$y - $label-size}" dy="{.6 * $label-size}"
                            text-anchor="middle">
115                         <xsl:value-of select="@relname"/>
116                     </text>
117                 </xsl:if>
118             </xsl:when>
119             <xsl:when test="rst:is_satellite(.)">
120                 <path d="M{$arc-x-pos},{$y} Q{($arc-x-pos + $arc-x-pos-p) div 2},{$y - .75 *
                        $span-height} {$arc-x-pos-p},{$y}" stroke-width="{rst:mm-to-px(.3527 *
                        $stroke-width-pt)}" fill="none" marker-end="url(#arrowhead-graph)"/>
121                 <text x="{($arc-x-pos + $arc-x-pos-p) div 2}" y="{$y - $label-size}" dy="{.6 *
                        $label-size}" text-anchor="middle">
122                     <xsl:value-of select="@relname"/>
123                 </text>
124             </xsl:when>
125         </xsl:choose>
126     </xsl:if>
127
128 </xsl:template>
129
130 <!-- returns span's ID -->
131 <xsl:function name="rst:get-node-no" as="xs:integer">
132     <xsl:param name="node" as="node()"/>
133     <xsl:value-of select="$node/count(preceding::text()[parent::rst:segment and normalize-space(.) !=
            ''] | (child::rst:*[not(rst:is_satellite(.))] |
            child::text())[1]/preceding-sibling::rst:*/descendant-or-self::rst:segment)"/>
134 </xsl:function>
135
136 <!-- returns span's ID range -->
137 <xsl:function name="rst:get-node-range" as="xs:integer">
138     <xsl:param name="node" as="node()"/>
139     <xsl:value-of select="$node/(count(descendant-or-self::rst:segment) -
            count(child::rst:*[rst:is_satellite(.)]/descendant-or-self::rst:segment))"/>
140 </xsl:function>
141
142 <!-- returns span's tree depth -->
143 <xsl:function name="rst:get-node-level" as="xs:integer">
144     <xsl:param name="node" as="node()"/>
145     <xsl:value-of select="$node/count((ancestor-or-self::rst:*)[not(rst:is_satellite(.))]) - 1"/>
146 </xsl:function>
147
148 <!-- returns visual span's x-position -->
149 <xsl:function name="rst:get-x-pos" as="xs:integer">
150     <xsl:param name="node" as="node()"/>
151     <xsl:value-of select="rst:get-node-no($node) * ($span-width + $span-margin) + $graph-margin"/>
152 </xsl:function>
153
154 <!-- returns visual span's y-position -->
155 <xsl:function name="rst:get-y-pos" as="xs:integer">
156     <xsl:param name="node" as="node()"/>
157     <xsl:value-of select="rst:get-node-level($node) * $span-height + $graph-margin"/>
158 </xsl:function>
```

```
159
160     <!-- returns visual span's width -->
161     <xsl:function name="rst:get-width" as="xs:integer">
162         <xsl:param name="node" as="node()"/>
163         <xsl:value-of select="rst:get-node-range($node) * ($span-width + $span-margin) - $span-margin"/>
164     </xsl:function>
165
166     <!-- returns visual span's arc x-position  -->
167     <xsl:function name="rst:get-arc-x-pos" as="xs:double">
168         <xsl:param name="node" as="node()"/>
169         <xsl:choose>
170             <xsl:when test="$node/count(rst:*[not(rst:is_satellite(.))]) &gt; 0">
171                 <xsl:value-of select="(rst:get-arc-x-pos($node/rst:*[not(rst:is_satellite(.))][1]) +
                        rst:get-arc-x-pos($node/rst:*[not(rst:is_satellite(.))][last()])) div 2"/>
172             </xsl:when>
173             <xsl:otherwise>
174                 <xsl:value-of select="rst:get-x-pos($node) + .5 * rst:get-width($node)"/>
175             </xsl:otherwise>
176         </xsl:choose>
177     </xsl:function>
178
179     <!-- suppress default text output -->
180     <xsl:template match="text()"/>
181
182     <!-- test if a given node is a RST nucleus (non-multinuclear) -->
183     <xsl:function name="rst:is_nucleus" as="xs:boolean">
184         <xsl:param name="node" as="node()"/>
185         <xsl:value-of select="$node/self::rst:*/not(@reltype)"/>
186     </xsl:function>
187
188     <!-- test if a given node is a RST satellite -->
189     <xsl:function name="rst:is_satellite" as="xs:boolean">
190         <xsl:param name="node" as="node()"/>
191         <xsl:value-of select="$node/self::rst:*/@reltype = 'rst'"/>
192     </xsl:function>
193
194     <!-- test if a given node is in multinuclear RST relation -->
195     <xsl:function name="rst:is_multinuclear" as="xs:boolean">
196         <xsl:param name="node" as="node()"/>
197         <xsl:value-of select="$node/self::rst:*/@reltype = 'multinuc'"/>
198     </xsl:function>
199
200 </xsl:stylesheet>
```

C.3. Genre Processing

C.3.1. Genre-specific Transformation Pipeline

genre.xpl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=============================================================================================================
    XProc pipeline to transform a genre-specific RST structure to its subgenre-specific co-structures
    @author   Frederik R.N. Schlupkothen
    @version  15.07.2015
    =============================================================================================================-->
<p:declare-step version="1.0"
    xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:c="http://www.w3.org/ns/xproc-step"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags">

    <p:import href="xlink-validate.xpl"/>

    <p:identity name="dag-input"/>

    <dag:validate-with-schematron>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="schema">
            <p:document href="genre.sch"/>
        </p:input>
    </dag:validate-with-schematron>

    <p:sink/>

    <p:xslt>
        <p:input port="source">
            <p:pipe port="result" step="dag-input"/>
        </p:input>
        <p:input port="stylesheet">
            <p:document href="genre.xsl"/>
        </p:input>
        <p:input port="parameters">
            <p:empty/>
        </p:input>
    </p:xslt>

</p:declare-step>
```

C.3.2. Standoff Markup Validation Pipeline

xlink-validate.xpl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=============================================================================================================
    XProc pipeline to validate a XLink-based standoff markup document structure via Schematron
    @author   Frederik R.N. Schlupkothen
    @version  15.07.2015
    =============================================================================================================-->
<p:declare-step version="1.0"
    xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:c="http://www.w3.org/ns/xproc-step"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
    type="dag:validate-with-schematron" name="main">

    <p:input port="source"/>
    <p:input port="schema"/>
    <p:output port="result"/>

    <p:import href="xlink-stdoff2final.xpl"/>

    <!-- transform standoff to inline document -->
    <dag:stdoff2inl name="source-inline">
        <p:input port="source">
            <p:pipe port="source" step="main"/>
        </p:input>
    </dag:stdoff2inl>

    <!-- resolve schema's URIs (included XSLT stylesheet's relative paths erroneous resolved in schema) -->
    <p:make-absolute-uris match="//xsl:include/@href" name="schema">
        <p:input port="source">
            <p:pipe port="schema" step="main"/>
        </p:input>
    </p:make-absolute-uris>

    <!-- validate inline document -->
    <p:validate-with-schematron>
        <p:input port="source">
            <p:pipe port="result" step="source-inline"/>
        </p:input>
        <p:input port="schema">
            <p:pipe port="result" step="schema"/>
        </p:input>
        <p:with-param name="allow-foreign" select="'true'"/>
    </p:validate-with-schematron>

</p:declare-step>
```

C.3.3. Embedded Standoff to Inline Markup Pipeline

stdoff-transform.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=============================================================================================================
    XSLT stylesheet describing an adaptable pipeline to transform XLink standoff markup to inline markup
    @author   Frederik R.N. Schlupkothen
    @version  10.07.2015
    ============================================================================================================-->
<xsl:stylesheet version="2.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dag="http://dmt.uni-wuppertal.de/schlupko/xlink4dags"
    exclude-result-prefixes="#all">

    <xsl:import href="xml-cleanup.xsl"/>
    <xsl:import href="xlink-inl2final.xsl"/>
    <xsl:import href="xlink-stdoff2inl.xsl"/>

    <xsl:param name="resolve-content-mode" select="'false'"/>


    <xsl:template match="/">
        <!-- transform standoff to intermediate inline markup -->
        <xsl:variable name="intermediate">
            <xsl:call-template name="process-stdoff"/>
        </xsl:variable>
        <!-- transform intermediate to final inline markup -->
        <xsl:variable name="final">
            <xsl:apply-templates select="$intermediate" mode="process-dag"/>
        </xsl:variable>
        <!-- transform final document structures -->
        <xsl:apply-templates select="$final/node()"/>
    </xsl:template>


<!--filter-nodes-->

<!---
    Process content nodes.
-->
    <xsl:template name="content-nodes">
        <xsl:apply-templates select="dag:alt"/>
    </xsl:template>


<!---
    Process structure nodes.
-->
    <xsl:template name="structure-nodes">
        <xsl:apply-templates select="*[not(self::dag:alt)]"/>
    </xsl:template>


```

```
<!---
    Select content nodes.
    @param node()  context  Context node of potential content nodes.
-->
    <xsl:function name="dag:content-nodes" as="node()*">
        <xsl:param name="context" as="node()"/>
        <xsl:sequence select="$context/dag:alt"/>
    </xsl:function>


<!---
    Select structure nodes.
    @param node()  context  Context node of potential structure nodes.
-->
    <xsl:function name="dag:structure-nodes" as="node()*">
        <xsl:param name="context" as="node()"/>
        <xsl:sequence select="$context/*[not(self::dag:alt)]"/>
    </xsl:function>


<!---
    Test if a given node is a content node.
    @param node()  context  Context node to be tested.
-->
    <xsl:function name="dag:is-content-node" as="xs:boolean">
        <xsl:param name="context" as="node()"/>
        <xsl:value-of select="not(empty($context/self::dag:alt))"/>
    </xsl:function>


<!---
    Test if a given node is a structure node.
    @param node()  context  Context node to be tested.
-->
    <xsl:function name="dag:is-structure-node" as="xs:boolean">
        <xsl:param name="context" as="node()"/>
        <xsl:value-of select="empty($context/self::dag:alt)"/>
    </xsl:function>


<!--process-content-->

<!---
    Transform content nodes, eventually wrapped, preceded, or followed by given XML nodes.
    @param xs:string  wrapper            The wrapping element's name (cf. XProc's p:wrap).
    @param xs:string  wrapper-namespace  The wrapping element's namespace.
    @param node()*    insert-before      The preceding node set.
    @param node()*    insert-after       The following node set.
-->
    <xsl:template match="dag:alt">
        <xsl:param name="wrapper" select="''" as="xs:string" tunnel="yes"/>
        <xsl:param name="wrapper-namespace" select="''" as="xs:string" tunnel="yes"/>
        <xsl:param name="insert-before" select="()" as="node()*" tunnel="yes"/>
        <xsl:param name="insert-after"  select="()" as="node()*" tunnel="yes"/>
        <xsl:copy-of select="$insert-before"/>
```

```
107         <xsl:choose>
108             <xsl:when test="$wrapper = ''">
109                 <xsl:apply-templates select="self::*" mode="process-dag"/>
110             </xsl:when>
111             <xsl:otherwise>
112                 <xsl:element name="{$wrapper}" namespace="{$wrapper-namespace}">
113                     <xsl:apply-templates select="self::*" mode="process-dag"/>
114                 </xsl:element>
115             </xsl:otherwise>
116         </xsl:choose>
117         <xsl:copy-of select="$insert-after"/>
118     </xsl:template>
119 
120 </xsl:stylesheet>
```

D. Compact Disc's Table of Contents

The following data is provided on the accompanying compact disc.

Directory and Files	Description	Appx.
examples/Fieldguide/	The "Fieldguide" example	B.2
examples/Fieldguide/doc.gannet/	The "Gannet" document	B.2.1
Gannet.basic_units.xhtml	– Content base's basic units	*p. 160*
Gannet.content.text.xhtml	– Content base's text realizations	*p. 166*
Gannet.content.list.xhtml	– Content base's list realizations	*p. 167*
Gannet.content.table.xhtml	– Content base's table realizations	*p. 168*
Gannet.content.img.xhtml	– Content base's image realizations	*p. 170*
Gannet.content.audio.xhtml	– Content base's audio realizations	*p. 171*
Gannet.content.video.xhtml	– Content base's video realizations	*p. 172*
Gannet.content.3d.xhtml	– Content base's spatial realizations	*p. 173*
Gannet.RST.struct.xhtml	– Document's RST structure	*p. 174*
Gannet.RST.labels.xhtml	– Document's RST labels	*p. 177*
Gannet.doc.xhtml	– Document's multistructure	*p. 182*
xhtml-xlink.dtd	– XHTML 1.1 plus XLink 1.1 DTD	A.4
examples/Fieldguide/doc.gannet/media/	Embedded media files	—
examples/Fieldguide/doc.gannet/media/audio/	Embedded audio files	—
examples/Fieldguide/doc.gannet/media/images/	Embedded image files	—
examples/Fieldguide/doc.gannet/media/video/	Embedded video files	—
examples/Fieldguide/doc.gannet/script/	Used script libraries	—
examples/Fieldguide/gen.fieldguide/	The "Fieldguide" genre	B.2.2
Fieldguide.content.sch	– Genre-specific RST key structures	*p. 183*
Fieldguide.content2form.print.xpl	– Print-specific XProc pipeline	*p. 184*
Fieldguide.content2form.print.xsl	– Print-specific XSL transformation	*p. 185*
Fieldguide.form2texml.print.xsl	– HTML to LaTeX transformation	—
Fieldguide.content2form.movie.xpl	– Movie-specific XProc pipeline	*p. 188*
Fieldguide.content2form.movie.xsl	– Movie-specific XSL transformation	*p. 189*
Fieldguide.content2form.3d.xpl	– 3D-specific XProc pipeline	*p. 196*
Fieldguide.content2form.3d.xsl	– 3D-specific XSL transformation	*p. 197*
examples/Fieldguide/out.gannet/	Subgenre-specific output	B.2.2
Gannet.doc.print.pdf	– Print-specific PDF output	*p. 187*
Gannet.doc.movie.html	– Movie-specific HTML output	*p. 193*

Directory and Files	Description	Appx.
Gannet.doc.3d.html	– 3D-specific HTML output	*p. 204*
Gannet.doc.3d.1.svg	– 3D-specific SVG output (poster 1)	*p. 206*
Gannet.doc.3d.2.svg	– 3D-specific SVG output (poster 2)	*p. 207*
implementation/	Implementation files	C
implementation/stdoff/	Multistructured document processing	C.1
xlink-traverse.xsl	– XLink traversal stylesheet	C.1.1
dag-functions.xsl	– XSL function library for DAGs	C.1.2
xlink-stdoff2inl.xsl	– Standoff to inline markup transf.	C.1.3
xlink-inl2final.xsl	– Intermediate to final markup transf.	C.1.4
xml-cleanup.xsl	– Namespace reorganization stylesheet	C.1.5
xlink-stdoff2inl.xpl	– Standoff to inline markup pipeline	—
xlink-stdoff2final.xpl	– Standoff to final markup pipeline	C.1.6
	Genre-specific processing	C.3
xlink-validate.xpl	– Standoff markup validation pipeline	C.3.2
stdoff-transform.xsl	– Standoff to inline markup XSLT pipel.	C.3.3
implementation/inl/	Multistructured document import	—
xlink-inl2stdoff.xsl	– Inline to standoff markup transf.	—
relpath_util.xsl	– XSL function library to resolve URLs	—
implementation/rst/	RST processing	C.2
RST.dtd	– Original flat RST DTD	C.2.1
deepRST.dtd	– Reformulated deep RST DTD	C.2.2
rst.sch	– Deep RST Schematron Schema	C.2.3
rst-functions.xsl	– XSL function library for RST	C.2.4
flat2deepRST.xsl	– Flat to deep RST transformation	C.2.5
deep2flatRST.xsl	– Deep to flat RST transformation	C.2.6
deepRST2SVG.xsl	– RST to SVG transformation	C.2.7
dag-functions.xsl	– XSL function library for DAGs	C.1.2
implementation/TeXML/	HTML to LaTeX transformation pipeline	—

E. Curriculum Vitae

Der Lebenslauf ist in der Online-Version aus Gründen des Datenschutzes nicht enthalten.