Content Structures, Organization, and Processes for Localized Content Management

Hans-Werner Sehring
Namics
Hamburg, Germany
e-mail: hans-werner.sehring@namics.com

Abstract—**Content management systems are in widespread use for document production. In particular, we see the pervasive application of web content management systems for web sites. These systems serve both authors that produce content and web site users that perceive content in the form of documents. Today, one focus lies on the consideration of the context of the web site user. Context is considered in order to serve users' information needs best. Many applications, e.g., marketing sites, focus on making the user experience most enjoyable. To this end, content is directed at the users' environmental and cultural background. This includes, first and foremost, the native language of the user. Practically, all respective web sites are offered in multiple languages and, therefore, multilingual content management is very common today. Content and its structure need to be prepared by authors for the different contexts, languages in this case. Contemporary content management system products, though, each follow different approaches to model context. There is no single agreed-upon approach because the different ways of modeling context put an emphasis on different content management properties. This paper discusses different aspects of multilingual content management and publication. The Minimalistic Meta Modeling Language is well suited for context-aware content management. This paper demonstrates how this modeling language can be used to master the requirements of multilingual content management within the framework of a common meta model. This allows content modeling without consideration of CMS product properties. This way, it takes away constraints from content modeling and it removes dependencies to content management system products.**

Keywords-content management; web site management; multilingual content management; multilinguality; localization; internationalization; context-awareness.

I. INTRODUCTION

Web sites are operated by nearly all organizations and enterprises, and they are created for various purposes. The management of such sites has evolved to *content management* that separates web site content, structure, and layout from each other. This way, content can be published on different media and on different channels. Certain parameters can influence the production of documents from content, e.g., the viewing device used by the user or her or his current context.

Consequently, most web sites are produced by a *content management system* (CMS), a web CMS in this case. Currently, web sites increasingly exhibit consideration of content that is tailored to the context of the content's percipient. They do so either to provide content with maximal value to the visitor, in order to inform a user in the most suitable way, to convey a message best, to present a company or a brand in the most appealing way, etc.

The most basic contextual property, to this end, is the native language of the consumer. Content should be presented to the user in this language. At least, textual content is translated. More advanced approaches take the culture and the habits of a user into account.

(Written) Language has an impact on layout. E.g., there need to be web page layouts for languages written from left to right and for ones written from the right to the left. On top of that, the writing direction has an impact on, e.g., the placement of navigation and search elements on a web page.

The relationship between translations of content can be viewed as contextualization as pointed out in an initial paper on the topic [1]. A translation is content to be managed in the context of the original content it has been derived from.

Some time ago, multilingual web appearances and print publications were identified as a major challenge for organizations [2]. In practice today, the problem is addressed by various approaches in different CMS products. However, there is no systematic consideration of these approaches and the characteristics of the resulting solutions. This leads to the content management approach taken to be dependent on the CMS product chosen for a particular web site appearance.

This paper reviews approaches to multilingual content management. The Minimalistic Meta Modeling Language allows abstract representations of these approaches and comparing their implications on content management tasks.

The rest of this paper is organized as follows. Section II defines requirements for multilingual web sites and the CMSs producing them. Section III describes related approaches to multilingual web site production. Section IV briefly introduces the Minimalistic Meta Modeling Language. Section V discusses the application of this language for multilingual content management. The conclusions and acknowledgement close the paper.

II. LOCALIZED CONTENT MANAGEMENT REQUIREMENTS

The general approach to multilingual web site production is similar for most approaches. It is a two-phase process.

Its foundation is language- and country-independent content, or at least content storage organized in a way that allows content to be localized easily. To this end, an initial *internationalization* (often abbreviated as I18n) step removes all cultural assumptions from content.

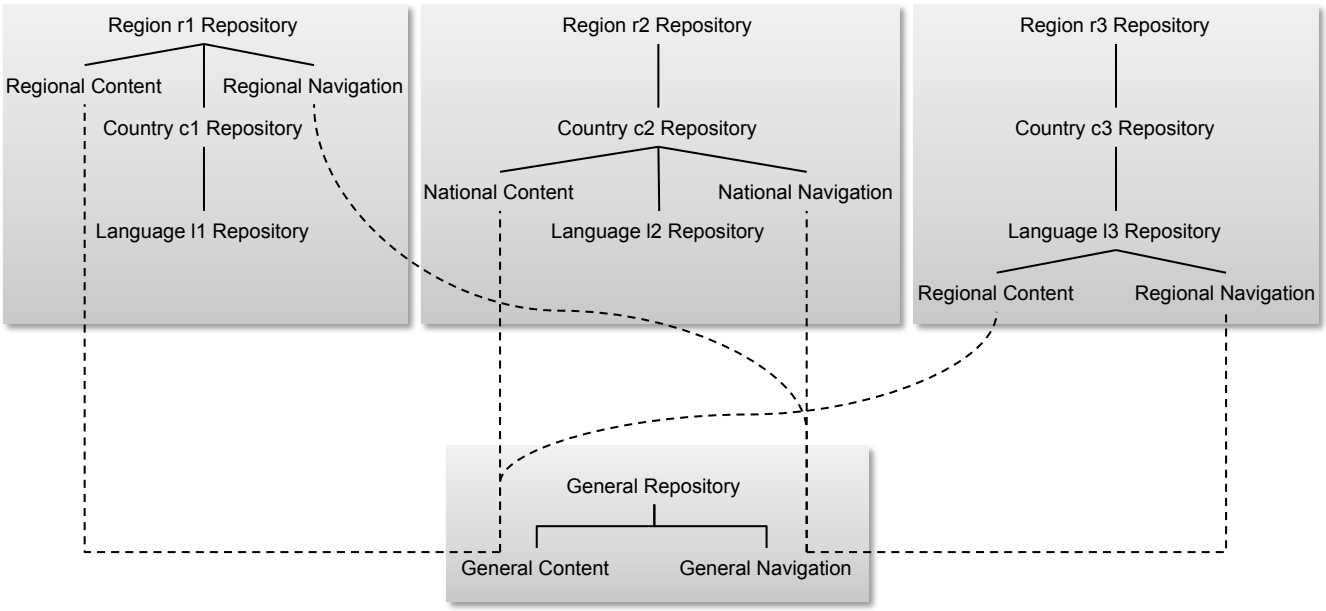

Figure 1. Example of a content repository organization for multilingual content with content distribution on different levels.

On that basis, a *localization* (L10n) procedure adapts content for a specific country, region, or language.

There are many considerations that have to be taken into account to enable this basic process. On top of the linguistic and cultural tasks of localization, there are business considerations and technical issues about the management of content, its structure, and organization (its physical structure) [3]. Choices are made based on the required content management strategy.

Content management processes for localization have to be defined. As part of those, the process of rendering localized content into publishable documents needs to be aligned with the actual content organization.

In this paper, we concentrate on the technical issues of multilingual content management and representation.

A. Basic Multilingual Content Management Strategies

There are three typical strategies for the management of multilingual and multicultural content [4]:

1) Central control over the content: Content is distributed by a central authority and it is translated to different target languages, but it is typically not adapted in other ways. I.e., there are no structural changes, and the layouts are not adapted to local preferences.

2) Decentralized management of multiple local sites without coordination: The local sites typically use a localized design. This approach does not ensure homogeneous quality in all localized appearances, there is no means to enforce content to be current in all local repositories, and there is no way to grant a globally recognizable web site standard, e.g., a corporate design.

3) A hybrid approach of the first two: It allows dealing with global, regional, and local content. Global content is produced centrally and translated for global use. Regional content is localized from centrally provided content, but is also adapted to and used in a regional context. Local content is produced locally in the local language in addition to global and regional content.

Because of the possibly combined advantages, many organizations favor the third approach. It requires tool support that is discussed in the subsequent subsections.

In practice, there are basically two CMS setups that correspond to centralized and decentralized content management: a central multi-tenant CMS that allows hosting multiple sites and relating these to each other, and isolated local CMSs that exchange content while providing their own web site structure and layout. The subsequent subsections of this section discuss these two approaches.

Fig. 1 shows an overview over different exemplary content repositories organized according to the two ways of multisite content management. Each repository represents one CMS instance or one content collection inside a CMS together with its structure, layouts, etc.

The three repositories at the top of Fig. 1 show content localization at different levels in content trees of multisite CMSs. In this example, each CMS hosts collections for regional, national, and language-specific content. These are just three arbitrary levels of content collections. The solid lines in the figure denote relationships between collections where the lower one is derived from the upper one.

The *General Repository* at the bottom represents the pool of internationalized content that is used for content distribution from a central content pool. The dashed lines represent content passed from one repository to another.

The sample repositories in Fig. 1 contain content (text, images, etc.), as well as navigation nodes and structure. These parts of the repositories are only shown where needed.

Maintaining content consistency across different localized versions is time consuming and error prone. There

are two primary ways of content distribution and localization: manual and semiautomatic [3]. These apply to both approaches, centralized as well as decentralized CMSs.

B. Multilingual Content Structures

Content needs a defined structure in order to be processed by content management systems. There are two basic forms of content organization for the support of the aforementioned multilingual content management strategies: related content variants and independent content collections.

1) Related content variants: Professional CMSs allow defining content collections and relating them to each other. A typical pattern is a master-variant model, where a *variant* can be derived from every piece of content. The original content then plays the role of a *master*. Typically the master is internationalized content in a certain language, and the variants are the translations. The structures of master and variants are identical or at least similar.

2) Independent content collections: In a decentralized approach, CMSs maintain local content and structures that are connected logically only. Localization may be performed by translation of internationalized content that is provided in a central content pool, by adding new local content, and by omitting centrally provided content from a local repository.

C. Content Localization Processes

The two ways of organizing content, either as related variants or as independent content collections, allow different localization processes. The most important ones are the translation of content from an internationalized form into a localized variant, and the transmission of content from a central content pool to a national, regional, or local content collection.

1) Translation of content using a master-variant model: Whenever the master content changes, some actions on the localized variants are induced. In terms of Fig. 1, e.g., when content in a regional repository is modified then localizations in the national repositories are updated.

Whenever master content is changed and variants are not yet translated, as well as in cases where the master is extended with additional content, e.g., new substructures, the variants have to be considered outdated. In these cases a repository may provide *default* or *fallback* values to the variants. In the simplest case the master content is taken as the fallback. Often the English version of a web site is chosen as a master, so that new content that is not yet localized shows up in English on the various sites.

Fallbacks are problematic for composite documents, e.g., images embedded in a text [5]. When an image is updated, this may, e.g., result in an English image contained in a French text. An application-specific fallback logic may be needed for composites.

Additionally, changes to the master may result in translation workflows being started. Such a workflow either demands that new or changed content is translated manually, or it employs automatic translation tools to create localized content. There are various degrees of *machine translation*, ranging from machine-aided human translation, over human-aided machine translation to fully automated translation [6].

It is current practice to minimize manual translation effort by using a *Translation Memory System* (TMS) that is employed for terminology management and to record all existing translations [7]. To enable these to work with a CMS, there are content interchange formats like the *XML Localisation Interchange File Format* (XLIFF) [8] and *Translation Memory eXchange* (TMX) [9] for output and input of multilingual content.

During automatic localization there are easy translation tasks like adaptations of number formats, measurement units, currencies, etc. From a cultural viewpoint, there is no general answer to the question whether a document's content can be changed while retaining its structure, though [10]. In general, only the translation of content according to a centrally given structure is achievable.

There are approaches to automatic translation that employ semantic models of content, e.g., ontology-based [11]. These are mainly subject to research.

In any case, it is crucial for editors to learn how to prepare content in a way that is suitable for localization [12].

2) Shipping of content between independent repositories: For the translation processes mentioned above, content needs to be exchanged between a CMS and a TMS using some external format that allows identifying related content when content is transferred back to the CMS.

When following a content strategy that employs independent content collections, content regularly needs to be transferred between repositories as part of the localization process. In terms of Fig. 1, e.g., internationalized content in the general repository is distributed to regional repositories.

For content collections inside one CMS, content transfer might just consist of internal references. If separate CMS instances need to exchange content, some external content format is required.

As indicated in Fig. 1, the repositories might form a hierarchical network of content pools, ranging from global over regional down to local repositories. Though these hierarchies result from the master-variant relationships (see Section B.1)), content shipping should take hierarchy levels into consideration (see the dashed lines in Fig. 1).

D. Content Presentation

Content is rendered into documents that are used for publication, e.g., HTML documents for web sites or XML documents for typesetting programs. Documents in various languages are produced from multilingual content.

Typically, predefined layouts are used for document creation. The way documents are created depends on the content structure, i.e., related content or independent content collections.

Related content is typically based on identical or at least similar structures. Rendering layouts for multilingual content have to be defined using a uniform structure. Consequently, a uniform layout or a set of layout variants is typically provided centrally. It is possible to provide local variations,

but this requires thorough knowledge of the content structure and its variants. While predefined layouts limit the degree to which presentations can be adapted to local preferences they support quality assurance. E.g., international enterprises with localized web sites typically wish to have their corporate identity and design (CI/CD) to be reflected in every local appearance.

The scenario where independent content collections are employed gives single CMS instances complete freedom concerning the visualization of content. Together with every decentralized repository, localized layouts can be defined. While this allows the highest degree of localization, it makes quality assurance harder. There is no enforced commonality between the local layouts.

III. Related Work

We briefly discuss related approaches to multilingual content management and commercial CMS products.

A. Modeling Approaches

Typically, the management of multilingual web sites relies on a CMS. There are approaches to solve multilingual content management on the level of HTML files, though.

MultiLingual XHTML (MLHTML) [13] is an extension to HTML. It was designed to include content for different languages in the same page file. An XSL style sheet is used to transform it to a plain HTML page for a given language. The approach is well suited for static sites without a CMS in the background and for large sets of existing static HTML pages. It requires web sites to have the same structure and the same layout across all languages, though.

B. Content Management System Products

Professional CMS products support multilingual content management. To name some examples, Adobe Experience Manager (AEM), CoreMedia CMS, and Sitecore Experience Platform all follow a master-variant approach to multisite management. They provide functionality to create a deep copy of a master site. The content entities from the copy are automatically related to the corresponding master entities.

All products allow local editing of content copies, and changes to the master lead to notifications sent to editors. CoreMedia also allows editing the content's structure. Sitecore manages navigation structures locally.

Some products add workflow tasks for the translation of all content entities. Workflows may drive automatic or manual translation processes. Some of the products provide workflows for external translations using XLIFF.

The master site serves as a fallback for missing localized content. To this end, AEM and CoreMedia allow to freely choose the master. Sitecore prefers US English for master content and it additionally provides fallback chains to, e.g., have a series of fallback languages before using the master.

IV. M3L

The *Minimalistic Meta Modeling Language* (*M3L*, pronounced "mel") is a modeling language that is applicable to a range of modeling tasks. It proved particularly useful for context-aware content modeling [14].

```
model          ::= ⟨def-list⟩
def-list       ::= ⟨def⟩ [⟨def-list⟩]
def            ::= ⟨ref⟩ "is" ⟨id-list⟩
                   ( "{"⟨def-list⟩"}" [⟨production-rule⟩]
                   | ⟨production-rule⟩
                   | ";"
                   )
ref            ::= ⟨id⟩ ["from" ⟨ref⟩]
id-list        ::= ("a" | "an" | "the") ⟨ref⟩ ["," ⟨id-list⟩]
production-rule ::= ( "|=" ⟨def⟩
                   | "|-" {⟨ref⟩ | string | "the" "name" }
                   ) ";"
```

Figure 2. Simplified grammer of the M3L.

In order to be able to discuss multilingual content management with M3L in the subsequent section, we briefly introduce the M3L modeling constructs.

M3L offers a rather minimalistic syntax that is described by the slightly simplified grammar (in EBNF) shown in Fig. 2. It is only simplified for readability and still complete.

The production for identifiers (*id*) is omitted here. It is a typical lexical rule that defines identifiers as character sequences. Identifiers may—in contrast to typical formal languages—be composed of any character sequence. Quotation is used to define identifiers containing whitespace, brackets, or other reserved symbols. The same holds for string literals (*string*).

The descriptive power of M3L lies in the fact that the formal semantics is rather abstract. There is no fixed domain semantics connected to M3L definitions. The exact semantics of M3L evaluation will not be discussed in this paper. For more details see [14].

A. Concept Definitions and References

A M3L model consists of a series of definitions (⟨*def*⟩ in the grammar definition above). Each definition starts with a previously unused identifier that is introduced by the definition and may end with a semicolon. For an example, see Fig. 3 line 1.

We call the entity referenced by such an identifier a *concept*. Its constituents are presented in the following.

The keyword `is` introduces an optional reference to a base concept. An inheritance relationship as known from object-oriented modeling is established between the base concept and the newly defined *derived concept*. This relationship leads to the concepts defined in the context (see below) of the base concept to be visible in the derived concept. Furthermore, the refined concept can be used wherever the base concept is expected (similar to subtype polymorphism).

As can be seen in the grammar, the keyword `is` always has to be followed by either `a`, `an`, or `the`. The keywords `a` and `an` are synonyms for indicating that a classification allows multiple sub concepts of the base concept. Lines 2 and 3 of Fig. 3 show an example.

There may be more than one base concept. Base concepts can be enumerated in a comma-separated list (Fig. 3 line 4).

```
001 NewConcept;
002 NewConcept is an ExistingConcept;
003 NewerConcept is an ExistingConcept;
004 NewConcept is an ExistingConcept, an AnotherExistingConcept;
005 TheOnlySubConcept is the SingletonConcept;

006 NewConcept;
007 NewConcept is an ExistingConcept;
008 NewConcept is an AnotherExistingConcept;

009 Person { name is a String; }
010 Peter is a Person { "Peter Smith" is the name; }
011 Employee { salary is a Number; }
012 Programmer is an Employee;
013 PeterTheEmployee is a Peter, a Programmer { 30000 is the salary; }

014 salary from Employee;

015 Person { sex; status; }
016 MarriedFemalePerson is a Person { female is the sex; married is the status; } |= Wife;
017 MarriedMalePerson is a Person { male is the sex; married is the status; } |= Husband;

018 Person { name is a String; } |- "<person>" name "</person>";
```

Figure 3. Code example of M3L statements.

The keyword `the` indicates a closed refinement: there may be only one refinement of the base concept (the currently defined one), e.g., line 5 of Fig. 3.

Any further refinement of the base concept(s) leads to the redefinition ("unbinding") of the existing refinements.

Statements about already existing concepts lead to their redefinition. E.g., the statements in lines 6-8 in Fig. 3 lead to the same definition of the concept *NewConcept* as the above variant.

Every statement defining a concept is also an expression that evaluates to a concept. Definition statements evaluate to the defined concept; e.g., line 6 of Fig. 3 has no effect on concept definitions, but it evaluates to *NewConcept* as defined in lines 1-4. Further evaluation rules follow from the remaining M3L constructs.

B. Content and Context Definitions

Concept definitions as introduced in the preceding section are valid in a *context*. Definitions like the ones seen so far add concepts to the topmost of a tree of contexts. Curly brackets open a new context. Lines 9-13 if Fig. 3 show an example.

In this example, we assume that concepts *String* and *Number* are already defined. The subconcepts created in context are unique specializations in that context only. In practice, the concept *30000* should also be given. If not, it will be introduced locally in the context of *PeterTheEmployee*, preventing reuse of the identical number.

M3L has visibility rules that correlate to contexts. Each context defines a scope in which definition identifiers are valid. Concepts from outer contexts are visible in inner scopes. E.g., in the above example the concept *String* is visible in *Person* because it is defined in the topmost scope. *salary* is visible in *PeterTheEmployee* because it is defined in *Employee* and the context is inherited. *salary* is not valid in the topmost context and in *Peter*. Contexts with those names may be defined later on, though.

Tying a context to a concept can be interpreted in different ways, e.g., as contextualization or as aggregation.

Contexts can be referenced using the projection operator `from` in order to use concepts across contexts. Fig. 3, line 14 shows an example where the salary of employee is selected.

C. Narrowing and Production Rules

M3L allows assigning one *semantic production rule* to each concept. Production rules fire when an instance comes into existence that matches the definition of the left-hand side of the rule. They replace the new concept by the concept referenced by the right-hand part of the rule.

In the example of Fig. 3 line 16, whenever a *MarriedFemalePerson* shall be created then a *Wife* is created instead.

Production rules are usually used in conjunction with M3L's *narrowing* of concepts. Before a production rule is applied, a concept is narrowed down as much as possible. Narrowing is a kind of matchmaking process to apply the most specific definition possible.

If a base concept fulfills all definitions—base concepts and constituents of the context—of a derived concept, then the base concept is taken as an equivalent of that derived concept. If a production rule is defined for the derived concept, this rule is used in place of all production rules defined for any super concept.

The code in lines 15-17 of Fig. 3 shows an example of combined narrowing and semantic production rules. Fig. 4 illustrates the narrowing of concepts resulting from these definitions. Whenever an "instance" (a derived concept) of *Person* is created, it is checked whether it actually matches one of the more specific definitions. A married female *Person* is replaced by *Wife*, a married male *Person* by *Husband*, every other *Person* is kept as it is.

In addition to the semantic production rules that create new concepts, M3L also has *syntactic production rules* like the one in line 18 of Fig. 2.

```
Person {                                    Person {
 male is the sex;                  →         male is the sex;
}                                           }

Person {
 female is the sex;
 married is the status;            →        Wife;
}

Person {
 male is the sex;
 married is the status;            →        Husband;
}
```

Figure 4. Illustration of the semantic production rules from Fig. 3.

Syntactic production rules evaluate to a string. The rules consist of a list of string literals and concept references whose production rules are applied recursively.

An additional keyword sequence `the name` (see grammar in Fig. 2) refers to the name of the concept to which the current rule belongs. This dynamic reference is required because syntactic production rules of a concept are chosen after narrowing in the same way as semantic production rules. In the example in Fig. 3 this means that the shown syntactic production rule may not only be applied to *Person*s, but also to refinements like *PeterTheEmployee*.

The syntactic rules are also used as grammar rules to generate recognizers that create concepts from strings.

If no rule is given, then the default syntactic production rule evaluates a concept to its name.

V. M3L FOR MULTILINGUAL CONTENT

To demonstrate how the M3L can be used to model multilingual content, we use a M3L representation of a setup like that from Fig. 1 as an example. Concepts for local repositories are derived from those of a central repository, this way relating content, content structure, navigation hierarchies, and document layouts. We briefly touch workflows and content interchange formats.

A. Content Models

We use M3L concepts to model content repositories and local collections as shown in Fig. 1 as well as to model content itself. Contextualization represents content structure. Relationships between repositories or collections are established by concept derivation.

In the course of the example, we concentrate on the navigation structure of a hypothetical website. In contrast to the actual content, this frees us from content modeling details that are not relevant for the discussion. The arguments to be discussed are the same in both cases.

We use contextualization for the navigation hierarchy. An example is shown in Fig. 5, lines 1-8. *ContentRepository*, *Content*, *Navigation*, and *NavItem* may be given concepts here. They are used to model repositories and collections, single content items, navigation hierarchies, and navigation entries, respectively.

In this example, the general repository is defined as a M3L concept holding concepts for centrally provided internationalized content. It encloses further concepts representing two main parts of the repository, the main content (*GeneralContent*) and the navigation structure (*GeneralNavigation*).

The navigation part hosts a central navigation structure with a main navigation node *Products+Services*. This one has subordinate navigation items *Consumer Products*, *Professional Products*, and *Support*.

The following models use derivation to relate translations of navigation items to those in the general repository.

We present two modeling alternatives to translate the navigation hierarchy. In the first alternative, outlined in Fig. 5, lines 9-16, editors translate each navigation item one by one. This way, the structure is kept as it is. We do so by deriving a sub concept, here *GermanNavigation*, from the general navigation. In this "copy" of the general navigation we can locally "replace" the navigation items by translations.

We provide exactly one translation (`is the`) per navigation item in the specific region context. Other translations can still be given in the context of other local repositories.

Changes in the general repository are propagated to local ones in such a model. E.g., when a new navigation item is added globally, it is inherited in the local repositories. Such an item will not be translated automatically, but the overall navigation structure stays up-to-date. The inherited concept provides a default value in this case.

As a second alternative, shown in Fig. 5 lines 17-24, we create a navigation structure locally. We populate it by picking single instances from the general repository. This way we detach the local structures from the global structure. The other properties, e.g., the pages assigned to a navigation node, are inherited, though.

The repository base is a new, "empty" one since *GermanNavigation* is derived from just *Navigation*, not *GeneralNavigation*. The inserted navigation items are derived from those from the global repository, though.

In such a detached repository, possible changes in the central repository are not propagated, but have to be reapplied locally. This can be performed either completely manually, or by means of a workflow (see below).

When structures are changed during localization, there are various possibilities for structural differences. The model in lines 25-37 of Fig. 5 gives two examples (without translation) for a company's web site in countries with smaller markets and, therefore, a smaller offering.

```
001 GeneralRepository is a ContentRepository {
002  GeneralContent is a Content { … }
003  GeneralNavigation is a Navigation {
004   "Products+Services" is a NavItem {
005    "Consumer Products" is a NavItem;
006    "Professional Products" is a NavItem;
007    Support is a NavItem;
008 } } }

009 GermanRepository1 is a GeneralRepository {
010  GermanContent is the GeneralContent { … }
011  GermanNavigation is the GeneralNavigation {
012   Produkte+Dienste is the Products+Services {
013    Verbraucher is the "Consumer Products";
014    Profis is the "Professional Products";
015    Kundendienst is the Support;
016 } } }

017 GermanRepository2 is a Repository {
018  GermanContent is a Content { … }
019  GermanNavigation is a Navigation {
020   Produkte+Dienste is a Products+Services from GeneralNavigation from GeneralRepository {
021    Verbraucher is a "Consumer Products" from GeneralNavigation from GeneralRepository;
022    Profis is a "Professional Products" from GeneralNavigation from GeneralRepository;
023    Kundendienst is a Support from GeneralNavigation from GeneralRepository;
024 } } }

025 NicheMarkets is a ContentRepository {
026  SmallCountry1 is a GeneralRepository {
027   Country1Content is the GeneralContent { … }
028   Country1Nav is the GeneralNavigation {
029    Products is the Products+Services {
030     "Consumer Products";
031     "Professional Products";
032  } } }
033  SmallCountry2 is a GeneralRepository {
034   Country2Content is the GeneralContent { … }
035   Country2Nav is the GeneralNavigation {
036    Products is the Products+Services;
037 } } }
```

Figure 5. Sample multilingual content model definitions using the M3L.

In the first example, *SmallCountry1*, a subset (two out of the three) of navigation items is inserted into the navigation tree below *Products*, the navigation item that generally appears as *Products+Services*. The second example, *SmallCountry2*, shows a flatter structure with no sub navigation items under *Products*.

Content and its structure are localized the same way as the navigation structure. Content typically is composed of multiple basic pieces of content, forming trees similar to the navigation hierarchies.

B. Layout Descriptions

Content is managed with the goal of finally being published. To this end, layouts are defined that are used to render content in documents.

Such documents can be published on certain *channels*. Channels are, in content management terms, media of a certain kind combined with means of document distribution. Examples are web sites, print publications, public kiosks, and mobile applications. Different channels require different presentations and include different sets of content.

Along with content also the layouts used for its publication can be localized when documents are produced using M3L's syntactic productions attached to localized concepts.

Fig. 6 shows an example of M3L code for localized layouts. Adding to the example of the repositories shown in Fig. 5, it presents two further repositories for Greek and French content.

In addition to the management of content alone, here a base concept *GeneralLayout* is given. This concept represents an additional part of content repositories holding layout descriptions (not shown in Fig. 1).

Inside this part of the repository, typical graphical elements like pages, text components, image components, etc. of the respective publication channel can be found. For this example, assume *Page* to be a given concept for web pages that aggregate content in one HTML file.

The main technical purpose of refinements of this concept is to define syntactic production rules that describe how documents are created from content. In the example of web pages, these rules create HTML and CSS code.

```
001 GreekRepository is a GeneralRepository {
002  GreekContent is the GeneralContent { … }
003  GreekNaviation is the GeneralNavigation {
004   NavItem is the NavItem { LatinTitle is a Title; }
005   …
006  }
007  GreekLayout is the GeneralLayout {
008   GreekPage is a Page
009   |- … "<html>" … "<body>" … GreekNavigation … "</body></html>";
010   GreekNavigation
011   |- "<ol class=\"greeknavigationbartoplevel\">"
012       "<li class=\"greeknavigationitem\">" NavItem "</li><ol>";
013   NavItem from GreekNavigation
014   |- "<ol class=\"greeknavigationbar\">"
015       "<li class=\"greeknavigationitem\">" the name
016        "<span class=\"latintranscription\">" LatinTitle "</span>"
017         NavItem "</li></ol>";
018   NavItem from Navigation
019   |- "";
020  }
021 }
022 FrenchRepository is a GeneralRepository {
023  FrenchContent is the GeneralContent { … }
024  FrenchNavigation is the GeneralNavigation { … }
025  FrenchLayout is the GeneralLayout {
026   FrenchPage is a Page {…}
027   |- … "<html>" … "<body>" … FrenchNavigation … "</body></html>";
028   FrenchNavigation
029   |- "<ol class=\"frenchnavigationbartoplevel\">"
030       "<li class=\"frenchnavigationitem\">" NavItem "</li><ol>";
031   NavItem from FrenchNavigation
032   |- "<ol class=\"frenchnavigationbar\">"
033       "<li class=\"frenchnavigationitem\">" the name
034         NavItem "</li></ol>";
035   NavItem from Navigation
036   |- "";
037  }
038 }
```

Figure 6. Sample multilingual layout definitions using the M3L.

For the course of the example we assume distinct layouts to be given for the two repositories. In Fig. 6, lines 7 to 20 sketch a fragment of a web page on the Greek web site, and the lines 25 to 37 show code for a French web page. In HTML it is typical to represent navigation hierarchies as nested ordered lists (ol) with list items (li) for leaf nodes.

Note that there are no references from the content or the navigation to the layouts. This way, multiple layouts can be defined for the same content, thus allowing *multi-channel publishing*.

All pages of our hypothetical web sites shall contain the whole navigation hierarchy of the web site they belong to. Therefore, the syntactic production rule for pages references the whole navigation part of the repository as the respective hierarchy's root.

The syntactic rule of *Navigation* outputs HTML code for the navigation root. As part of the production it addresses *NavItem*. This reference evaluates to the set of all contained (refinements of) *NavItem*s. This leads to all syntactic representations of the concrete *NavItem* refinements to be concatenated in the output.

*NavItem*s emit code for one navigation entry each. This includes their (localized) name. The hierarchy is traversed by recursion through the reference to the nested *NavItem* refinements.

In the case of leaf navigation items, there is no refinement of *NavItem*. For these, the reference evaluates to *NavItem* as defined in *Navigation*, not any of the refinements in the context of the localized repositories. We add a rule to the "plain" *NavItem* to terminate recursion (lines 18/19 and 35/36 in Fig. 6). Otherwise, lines 17 and 34 would print "NavItem" as the default production rule would be used.

In the example, the two sets of web pages mainly differ in some CSS classes that are attributed to HTML elements. To illustrate possible structural differences, Greek navigation items that are assumedly labeled in Greek writing contain an additional transcription using Latin letters.

In fact, this structural difference even has an impact on the content (the navigation, in our example). The Latin transcription needs to be maintained by content editors. Therefore, according content *LatinTitle* is defined in the context of every Greek navigation item by refining *NavItem* in that context.

With this redefinition Greek, navigation items also print the Latin transcription as part of the syntactic production rule for the layout (line 16 of Fig. 6).

```
001 WorkflowTask is a … { Agent is a …; }

002 TranslationWorkflowTask is a WorkflowTask {
003  ContentToTranslate is a String;
004  ResultingContent is a String;
005  Translator is the Agent;
006 } |= TranslatedContent;

007 NewsContent is a … {
008  Title is a String;
009  Text is a String;
010 }
011 GeneralNews is a NewsContent, an InternationalizedContent
012 |= TranslationWorkflowTask {
013  Title is a ContentToTranslate;
014  Text is a ContentToTranslate;
015  … Translator;
016 }
017 |- … (code exporting content, e.g. XLIFF, for the given translator)
018 News is a NewsContent, a TranslatedContent {
019  Title is the Title from TranslatedContent from Translator;
020  Text is the Text from TranslatedContent from Translator;
021 }
022 |- … (rule for importing content from, e.g., XLIFF)
```

Figure 7. Code example of basic M3L concepts for workflow definitions.

The example shown in Fig. 6 exhibits a lot of duplicate code. Note that in practical cases there will be reuse of layouts by providing standard layouts on the level of *GeneralRepository* or some additional intermediate repositories, rather than defining everything inside the local repositories. Such an intermediate repository might be, e.g., a repository for all languages using Latin writing.

The termination rule stating that *NavItem* renders an empty string may even be defined in the context of *GeneralRepository*, thus holding for every repository.

C. Workflows

Translation tasks in a CMS are often driven by workflows. Introducing a complete workflow management system is beyond the scope of this paper. We provide a sketch of an approach based on M3L structures.

A workflow consists of workflow tasks, e.g., represented by derivations of a *WorkflowTask* as shown in Fig. 7, line 1.

A translation workflow task derived from it may look like shown in Fig. 7, lines 2-6.

For the sake of simplicity, we assume content to consist of *String*s in this example. In practice, it may be structured.

We create workflows by deriving specific workflow tasks and by connecting them using semantic production rules. There are rules for content that initializing workflow tasks and rules for workflow tasks creating a subsequent step.

In the case of interactive workflow tasks, syntactic rules create representations for exchange with external processors.

The example in lines 7-17 of Fig. 7 shows a definition of content of type *GeneralNews*. Whenever new content that is derived from *GeneralNews* is created, its semantic rule is inherited and thus a *TranslationWorkflowTask* is created. *InternationalizedContent* contains content that needs to be localized and, therefore, starts a translation workflow task and initializes it with the *Title* and *Text* as content that needs translation. Inside a translation workflow task, *ContentToTranslate* flags content as requiring translation.

As a parameter directing the translation process, Fig. 7 shows a *Translator* in the context *TranslationWorkflowTask*. In practice, it may be evaluated by the following workflow execution.

The workflow tasks' syntactic production rules produce an external representation that can be passed to, e.g., a TMS.

When a result from such a translation service is received, a *TranslatedContent* is produced from it. We need this concept to distinguish it from *InternationalizedContent* so that is not subject to further workflow enactments.

In this example, the workflow yields the translated *NewsContent*. In order to create the concepts from translated external content, refinements like *News* in the example of Fig. 7 (lines 18-22) declare the mapping of an external representation to a M3L concept by a semantic rule.

D. Content Exchange

When sharing content with external parties, as discussed above, as well as in manual translation processes, content needs to be shipped between the different parties.

Inside one organization, communication can be established using M3L's structures directly. In order to interchange content with external organizations, we use an external format for input and output. This can be defined using M3L's syntactic production rules. Fig. 8 shows a sketch of an example.

In lines 1-4 of Fig. 8, the *Text* component of content of type *NewsContent* (compare Fig. 7) is externalized in XLIFF by means of the syntactic rule of *ExternalizableNews*. The resulting file can be sent to a translator, and the result can be parsed in to form an *ExternalizableNews* again.

```
001 ExternalizableNews is NewsContent
002 |- "<xliff …> … <source>"
003    Text
004    "</source> … </xliff>";
005 News is an ExternalizableNews;
006 GeneralNews is an ExternalizableNews;
```

Figure 8. Code example for a content interchage format.

In addition to Fig. 7, we also define *GeneralNews* and *News* as refinements of *ExternalizableNews* in order to inherit the syntactic rule (lines 5 and 6 of Fig. 8), replacing the sketch shown there.

With these definitions, *GeneralNews* can be exported in XLIFF using the syntactic rule for the production of XML. *News* can be imported from XLIFF using the syntactic rule to recognize XML.

VI. Summary and Outlook

This section recaps the paper and discusses future work.

A. Summary

Multilingual content management is in widespread use, and various requirements for the management of localized variants of global content exist. This paper discusses an approach to multilingual content management using context.

The Minimalistic Meta Modeling Language (M3L) is a general-purpose modeling language that has proven particularly useful for context-aware content management. In this paper we demonstrate how to employ M3L to model multilingual content management in a product-agnostic way. This way, properties of content management systems can be discussed independent of implementations.

B. Outlook

M3L can be executed by evaluating M3L statements. However, this kind of execution is not an adequate approach for building running systems. CMS products, on the other hand, are of practical importance. Therefore, in the future we want generate product configurations out of M3L statements.

To this end, software artifacts (e.g., configuration files and source code files) can be created using syntactic rules the way HTML is generated in the examples above. Alternatively, product-specific model compilers for content models [15] can possibly be adapted to M3L.

By means of generation, heterogeneous systems can be achieved by generating code for different CMS products from the same content model. This way, local repositories are free to choose an implementation technology.

The workflows for content localization need further work, in particular those incorporating external translators.

So far, internationalized content immediately creates a workflow task. In practice, sets of concepts are translated together. Therefore, a practical workflow needs to be enacted at a defined point in time on a selected content set.

Acknowledgment

The author wishes to express gratitude to his employer, Namics, for enabling him to follow his scientific ambitions.

The insights presented here are taken from numerous practical projects. Thanks to all the colleagues as well as the business and technology partners that helped understanding problems and developing ideas for their solution.

The anonymous reviewers of the original conference paper as well as those of this article are acknowledged for their valuable input that helped improving it.

References

[1] H.-W. Sehring, “Localized Content Management with the Minimalistic Meta Modeling Language,” Proc. The Ninth International Conference on Creative Content Technologies (CONTENT 2017), pp. 8-13, Feb. 2017.

[2] S. Mescan, “Why Content Management Should Be Part of Every Organization’s Global Strategy,” Information Management Journal, vol. 38, no. 4, pp. 54-57, Jul./Aug. 2004.

[3] S. Huang and S. Tilley, “Issues of Content and Structure for a Multilingual Web Site,” Proc. 19th Annual International Conference on Computer Documentation (SIGDOC '01), ACM New York, NY, USA, pp. 103-110, Oct. 2001.

[4] R. Lockwood, “Have Brand & Will Travell,” Language International, vol. 12, no. 2, pp. 14-16, 2000.

[5] J.-M. Lecarpentier, C. Bazin, and H. Le Crosnier, “Multilingual Composite Document Management Framework For The Internet: an FRBR approach,” Proc. 10th ACM Symposium on Document Engineering (DocEng '10), ACM New York, NY, USA, pp. 13-16, Sep. 2010.

[6] W. J. Hutchins and H. L. Somers, An Introduction to Machine Translation. Academic Press, 1992.

[7] E. Macklovitch and G. Russell, “What’s Been Forgotten in Translation Memory,” Proc. The 4th Conference of the Association for Machine Translation in the Americas on Envisioning Machine Translation in the Information Future (AMTA '00), pp. 137-146, Oct. 2000.

[8] Organization for the Advancement of Structured Information Standards (OASIS). XLIFF Version 2.0. [Online]. Available from: http://docs.oasis-open.org/xliff/xliff-core/v2.0/xliff-core-v2.0.html

[9] Globalization & Localization Association (GALA). TMX 1.4b Specification. [Online]. Available from: https://www.gala-global.org/node/59010

[10] P. Sandrini, “Website Localization and Translation,” Proc. EU High Level Scientific Conferences, Marie Curie Euroconferences, MuTra: Challenges of Multidimensional Translation, pp. 131-138, May 2005.

[11] D. Jones, A. O’Connor, Y. M. Abgaz, and D. Lewis, “A Semantic Model for Integrated Content Management, Localisation and Language Technology Processing,” Proc. 2nd International Conference on Multilingual Semantic Web (MSW'11), vol. 775, pp. 38-49, 2011.

[12] R. Miller, “Multilingual Content Management: Found in Translation,” EContent, vol. 29, no. 6, pp. 22-27, Jul. 2006.

[13] P. Tonella, F. Ricca, E. Pianta, and C. Girardi, “Restructuring Multilingual Web Sites,” Proc. International Conference on Software Maintenance, pp. 290-299, Oct. 2002.

[14] H.-W. Sehring, “Content Modeling Based on Concepts in Contexts,” Proc. The Third International Conference on Creative Content Technologies (CONTENT 2011), pp. 18-23, Sep. 2011.

[15] H.-W. Sehring, S. Bossung, and J.W. Schmidt, “Content is Capricious: A Case for Dynamic System Generation,” Proc. 10th East European Conference, ADBIS 2006, pp. 430-445, September 2006.