Localized Content Management with the Minimalistic Meta Modeling Language

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 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 multilingual content management have different focuses. 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 support a universal approach to multilingual content management. This approach allows content modeling without consideration of 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; 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. 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 convey a message best, to present a company in the best possible 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.

Some time ago, multilingual web appearances and print publications were identified as a major challenge for organizations [1]. 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.

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.

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.

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) [2]. Content management processes for localization have to be defined. In this paper, we concentrate on the technical issue of multilingual content representation.

A. Basic Multilingual Content Management Strategies

There are three typical strategies for the management of multilingual and multicultural content [3]:

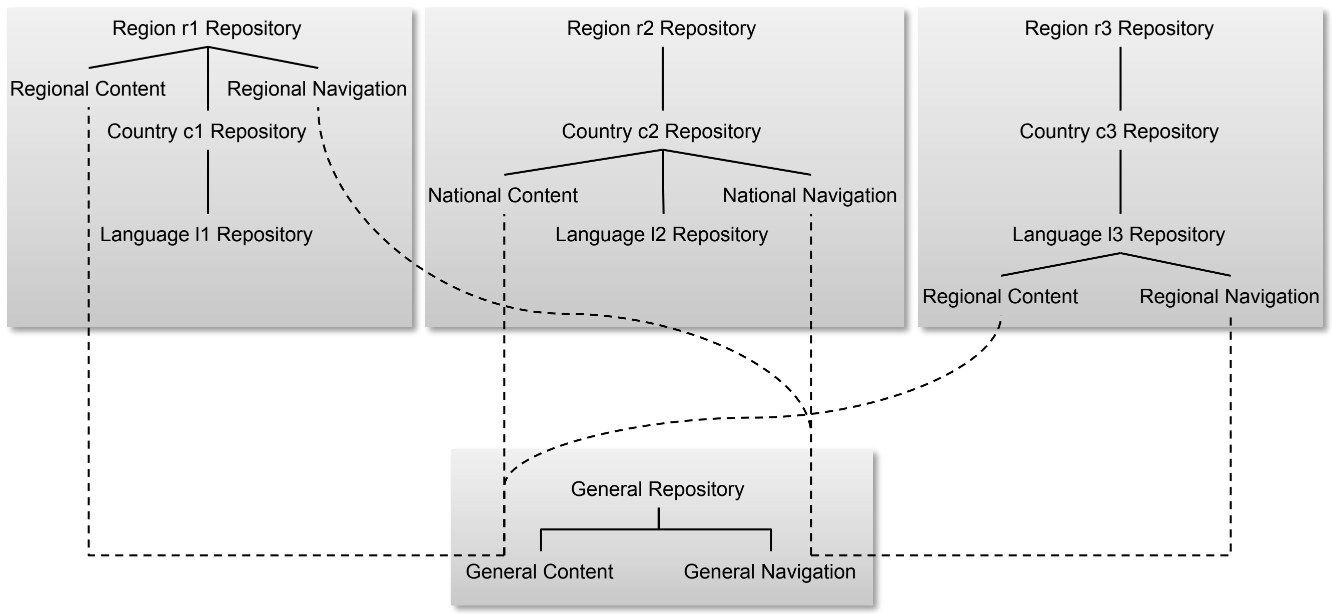

Figure 1. Example of a content repository organization for multilingual content with content distribution on different levels.

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 [2]. These apply to both approaches, centralized as well as decentralized CMSs.

B. Related Content

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*. Whenever the master content changes, some actions on the variants are induced.

When the master is extended with additional content, e.g., new substructures, then it may provide *default* or *fallback* values to the variants. 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 [4]. 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.

Manual translation typically has to be performed by professional translators. To enable these to work with a CMS, there are content interchange formats like the XML Localisation Interchange File Format (XLIFF) and Translation Memory eXchange (TMX) 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 [5]. In general, only the translation of content according to a centrally given structure is achievable.

Scientific approaches to automatic translation are based on semantic models of content, often ontology-based [6].

In any case, it is crucial for editors to learn how to prepare content in a way that is suitable for localization [7].

C. *Independent Content*

In a decentralized approach, CMSs maintain local content and structures. 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.

This scenario furthermore gives single CMSs complete freedom concerning the visualization of content.

Since a central repository provides base content, the decentralized approach requires means to ship content from that central instance to the local repositories. For content collections inside one CMS, the shipping 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 previous subsection), content shipping should take hierarchy levels into consideration (see the dashed lines in Fig. 1).

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) [8] 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 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. Instead, Sitecore 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 [9].

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 following slightly simplified grammar (in EBNF):

```
model           ::= ⟨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⟩]
def-list        ::= ⟨def⟩ [⟨def-list⟩]
production-rule ::= ("|=" ⟨def⟩ | "|-"{⟨ref⟩|string}) ";"
```

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 semantics of M3L evaluation will not be discussed in this paper. For more details see [9].

A. Concept Definitions and References

A M3L definition 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, e.g.:

```
NewConcept;
```

We call the entity referenced by such an identifier a *concept*.

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:

```
NewConcept is an ExistingConcept;
NewerConcept is an ExistingConcept;
```

There may be more than one base concept. Base concepts can be enumerated in a comma-separated list:

```
NewConcept is an ExistingConcept,
              an AnotherExistingConcept;
```

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

```
TheOnlySubConcept is the SingletonConcept;
```

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 following expressions lead to the same definition of the concept `NewConcept` as the above variant:

```
NewConcept;
NewConcept is an ExistingConcept;
NewConcept is an AnotherExistingConcept;
```

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 the topmost of a tree of contexts. Curly brackets open a new context, e.g.:

```
Person { name is a String; }
Peter is a Person{"Peter Smith" is the name;}
Employee { salary is a Number; }
Programmer is an Employee;
PeterTheEmployee is a Peter, a Programmer {
 30000 is the salary; }
```

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:

```
salary from Employee.
```

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.

The following shows an example:

```
Person {
 female is the sex; married is the status;
} |= Wife;
```

Whenever a `female Person` who is `married` 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 following code shows an example of combined narrowing and semantic production rules:

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

There is a concept `Person`. 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:

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

In addition to the semantic production rules that create new concepts, M3L also has *syntactic production rules*:

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

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.

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 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 the setup from Fig. 1. Local models are derived from a central repository, and we briefly touch workflows and content interchange formats.

A. Content Models

We use concepts to model repositories, local collections, and content as shown in Fig. 1. Their contextualization represents content structure. Relationships between repositories or collections are established by derivation.

For the example, we concentrate on the navigation structure. This frees us from content modeling details that are not relevant for the discussion.

We use contextualization for the navigation hierarchy:

```
GeneralRepository is a ContentRepository {
 GeneralContent is a Content { … }
 GeneralNavigation is a Navigation {
  "Products+Services" is a NavItem {
   "Consumer Products" is a NavItem;
   "Professional Products" is a NavItem;
   Support is a NavItem;
} } }
```

ContentRepository, *Content*, *Navigation*, and *NavItem* may be given concepts here.

In this example, the general repository hosts a central navigation structure with a main navigation node *Products+Services*. It 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, 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, e.g., *GermanNavigation* from the general navigation. In this "copy" of the general navigation we can locally "replace" the navigation items by translations.

```
GermanRepository is a GeneralRepository {
 GermanContent is the GeneralContent { … }
 GermanNavigation is the GeneralNavigation {
  Produkte+Dienste is the Products+Services {
   Verbraucher is the "Consumer Products";
   Profis is the "Professional Products";
   Kundendienst is the Support;
} } }
```

We provide exactly one translation (`is the`) per navigation item in the specific region context. In other contexts other translations can be given.

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.

As a second alternative 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.

Building *GermanNavigation* this way results in:

```
GermanRepository is a Repository {
 GermanContent is a Content { … }
 GermanNavigation is a Navigation {
  Produkte+Dienste is a Products+Services
   from GeneralNavigation
    from GeneralRepository {
   Verbraucher is a "Consumer Products"
    from GeneralNavigation
     from GeneralRepository;
   Profis is a "Professional Products"
    from GeneralNavigation
     from GeneralRepository;
   Kundendienst is a Support
    from GeneralNavigation
     from GeneralRepository;
} } }
```

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 following model gives two examples (without translation) for a company's web site in countries with smaller markets and, therefore, a smaller offering:

```
NicheMarkets is a ContentRepository {
 SmallCountry1 is a GeneralRepository {
  Country1Content is the GeneralContent { … }
  Country1Nav is the GeneralNavigation {
   Products is the Products+Services {
    "Consumer Products";
    "Professional Products";
 } } }
 SmallCountry2 is a GeneralRepository {
  Country2Content is the GeneralContent { … }
  Country2Nav is the GeneralNavigation {
   Products is the Products+Services; }
} }
```

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. We limit the example to navigation items in order to reduce its complexity.

Along with the content also the layouts used for its publication can be localized when documents are produced using M3L's syntactic productions. E.g., the production rules can generate HTML pages. In a very simple way:

```
GreekRepository is a GeneralRepository {
 GreekContent is the GeneralContent {
  GreekPage is a Page {…} |- … (Greek layout)
}
FrenchRepository is a GeneralRepository {
 FrenchContent is the GeneralContent {
  FrenchPage is a Page {…} |- … (French layout)
}
```

B. *Workflows*

Typically, translation tasks are 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:

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

Then a translation workflow task may look like:

```
TranslationWorkflowTask is a WorkflowTask {
 ContentToTranslate is a String;
 ResultingContent is a String;
 Translator is the Agent;
} |= TranslatedContent;
```

For the sake of simplicity we assume content to consist just of *String*s.

When the task is completed, it evaluates to a *TranslatedContent*. We need this type to distinguish it from *InternationalizedContent* (s.b.) that is processed in workflows.

We connect content to workflows by means of semantic production rules. E.g., the following example shows a definition of content of type *News* with a production rule creating a workflow task.

```
News is a TranslatedContent;
GeneralNews is an InternationalizedContent {
 Title is a String; Text is a String;
} |= TranslationWorkflowTask {
 Title is a ContentToTranslate;
 Text is a ContentToTranslate;
 … Translator;
} |= News { … };
```

Therefore, whenever new content that is derived from *GeneralNews* is created, the rule is inherited and thus a *TranslationWorkflowTask* is created, initialized with the *News*' *Title* and *Text* as content that needs translation. It yields the translated *News*.

C. *Content Exchange*

In manual translation processes, content needs to be shipped between 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. Example:

```
Content News { … } |- "<xliff …> … <source>"
                  Text "</source> … </xliff>"
```

Here, the *Text* component of content of type *News* is externalized in XLIFF. The resulting file can be sent to a translator, and the result can be parsed in to form a *News*.

VI. SUMMARY AND OUTLOOK

This section recaps the paper and discusses future work.

A. *Summary*

Multilingual content management is in widespread use, and requirements for the management of localized variants of global content can be formulated. 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.

B. *Outlook*

M3L can be executed by evaluating M3L statements. However, this 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 to build product-specific model compilers that generate product configurations out of M3L statements.

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

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.

REFERENCES

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.