Content Modeling Based on Concepts in Contexts

Hans-Werner Sehring
Content and Collaboration Solutions
T-Systems Multimedia Solutions GmbH
Dresden, Germany
Email: hans-werner.sehring@t-systems.com

***Abstract*—Content management is a generic term for different tasks dealing with content modeling, the creation and management of content instances, and the delivery of content as part of applications or documents. We studied content models for real-world entities in the area of Concept-oriented Content Management. In addition to content modeling, there is agreement that content heavily depends on context in most cases. Therefore, content management has to devise ways to consider context in its models and processes. In this paper we apply a novel generic content modeling language to address the modeling demands arising from content management that is augmented with context information.**

***Keywords-content management*; *content modeling*; *context*; *contextualization*; *personalization*.**

I. Introduction

Content management is a generic term for different tasks dealing with content modeling, the creation and management of content instances, and the delivery of content as part of applications or documents.

The definition of content is manifold. In particular, there are two basic notions of content: (1) (purely) digital content that is an entity of its own, and (2) content that is used to describe real-world entities. The latter employment of content is used for entities that cannot adequately be represented by structured data alone. One class of such content are product descriptions as found in product information management (PIM) applications. Catalogs seek to visualize products in an attractive way. Another class are complex entities, in particular ones from non-technical domains that do not rely on formal representations. In those domains content often represents (states of) a process, e.g., the inception, creation, and use of a work of art in art history.

We study content models for such real-world entities in the area of *Concept-oriented Content Management* (*CCM*) [6]. This paper presents a generic content modeling language to address the modeling demands from that area.

There seems to be agreement that content depends on *context* in most cases. This insight recently starts to have an increasing impact on content industry. Therefore, content management has to devise ways to consider context in its models, functions, and processes.

In this paper, we present a modeling language that allows combining different modeling approaches, including content description, classification, and contextualization.

The paper is organized as follows: In Section II, we formulate requirements for content models that describe entities in contexts. Section III defines a modeling language applicable for this kind of content representation. How typical modeling tasks are solved by that language is discussed in Section IV. The paper closes with conclusions in Section V.

II. Content Modeling Requirements

Content management requirements as laid out in the introduction demand for adequate content structures. To this end, content modeling is of central importance for content management applications. Typical content modeling approaches are discussed in this section.

A. Content Description

A range of modeling concepts and languages for the description of content has emerged. Figure 1 gives an overview over the most commonly used modeling techniques.

Digital content itself, from a technical point of view, simply consists of binary data representing a text, an image, a sound, etc. By adding descriptive information such data is enriched in order to be perceivable as content.

The basic level of content management is established by *meta data* and *descriptive information* that further describe the data or the entity represented by it, respectively. *Classification*, e.g. by tagging, can be seen as a specific kind of descriptive information that enables additional functionality like filtering and clustering. Classification is particularly useful if classifiers are related to each other, e.g. by narrower term and broader term relationships. Such relationships are provided by *taxonomies* and *ontologies*, e.g. defined explicitly or by description logic expressions, or freely assigned *tags* from which *folksonomies* are derived.

In CCM we also investigate *extensional descriptions* of content by providing a sample set for terms as epistemic structures. As an example three images describing the concept “strength” are shown in Figure 1 on the left-hand side. A similar approach is taken by automatic classification systems that take instances as a training set.

Many content management approaches rely on schema definitions that are based on the object-oriented paradigm. Though object-orientation offers many beneficial features

 ISBN: 978-1-61208-157-1

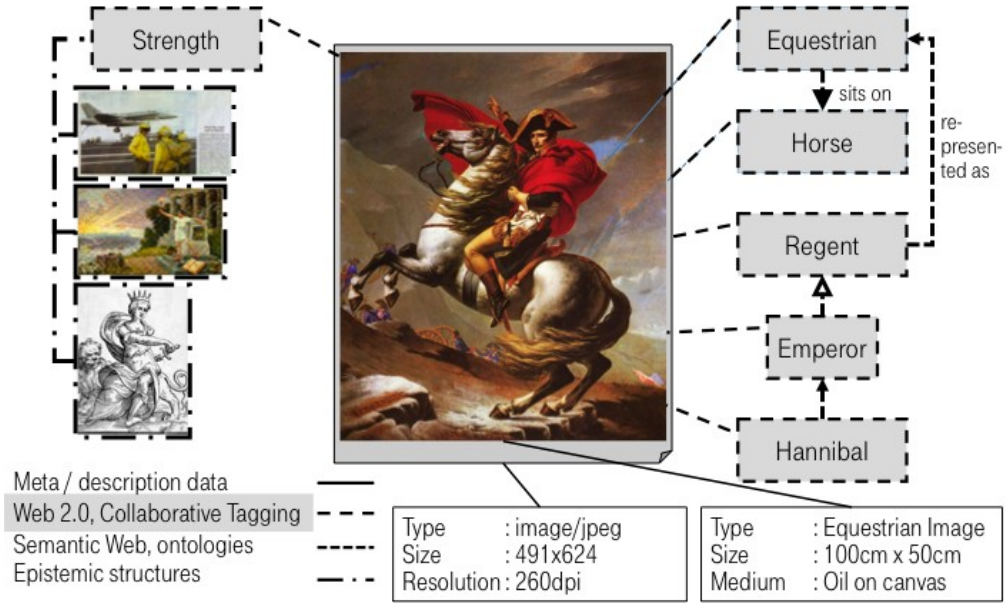

Figure 1. Content modeling approaches.

like inheritance and polymorphism, it is in several respects not well suited for those tasks.

In particular, the rigid distinction between classes and instances often is an obstacle for content modeling. Classes are supposed to provide a "blueprint" for instances. Instances are not supposed to deviate from the given form. This hinders the modeling of, e.g., variants, roles, and materializations that both represent views on one specific instance.

Furthermore, object-orientation is based on one single class hierarchy. In content management one typically has to deal with classes of different kinds. One main distinction is the one between structural definitions ("a book has a title and an author") and domain-specific classification information ("there are books on computer science and on mathematics"). Typically, the different kinds of classification exhibit different characteristics [4].

This paper presents a novel language with properties that make it suitable for content modeling. It borrows from the object-oriented paradigm, but it abstracts from certain concepts. This way it provides a basis for an integration of the above-mentioned modeling techniques.

B. Content Contextualization

In addition to content descriptions as provided by the approaches sketched in the previous section, the notion of context is of importance for content. There is agreement in the content management community that context is central to the task outlined in the introduction ("if content is king, context is the kingdom" [3]). Various studies underline the importance of context for content management [1].

The use of content is heavily influenced by the context of the user and the circumstances under which content was created. The context of content has to be considered with respect to modeling, interpretation, and delivery of content.

By introducing context into content models the definition of content variants as well as of the history of content creation and use is supported.

Context enables an extended content interpretation, with practical applications for content analysis and retrieval as well as the computation of recommendations. The value of content for a specific use can be judged if the current work context of the user is known, as well as the history of a particular piece of content by means of data provenance.

Modern content-based applications include a context-dependent delivery. Knowledge about the context of the user allows a personalized content presentation. For delivery the context often also refers to the publication channel and the device on which documents are displayed.

III. Minimalistic Meta Modeling Language

The *Minimalistic Meta Modeling Language* (*M3L*, pronounced "mel") is a modeling language that is currently under development and that has not been reported about yet. Though its further development is not particularly directed at content modeling, the rationale behind the language is based on the modeling tasks discussed in this paper.

M3L offers a rather minimalistic syntax that is completely covered by the following grammar (in BNF):

$$
\begin{aligned}
\textit{model} &::= \langle \textit{prop-list} \rangle \\
\textit{def} &::= \langle \textit{id} \rangle \text{“}\textit{is}\text{”} \langle \textit{id-list} \rangle \\
&\quad [\text{“}\{\text{”} \langle \textit{prop-list} \rangle \text{“}\}\text{”} [\langle \textit{production-rule} \rangle] \\
&\quad | \langle \textit{production-rule} \rangle | \text{“;”}]
\end{aligned}
$$

$$
\begin{aligned}
\textit{ref} &::= \langle \textit{id} \rangle [\text{“}\textit{from}\text{”} \langle \textit{id} \rangle] \\
\textit{id-list} &::= (\text{“}\textit{a}\text{”} | \text{“}\textit{an}\text{”} | \text{“}\textit{the}\text{”}) \langle \textit{ref} \rangle [\text{“,”} \langle \textit{id-list} \rangle] \\
\textit{prop-list} &::= \langle \textit{def} \rangle [\langle \textit{prop-list} \rangle] \\
\textit{production-rule} &::= \text{“}|-\text{”} \langle \textit{ref} \rangle \text{“;”} \\
\textit{id} &::= \ldots \textit{(regular expression of identifiers)}
\end{aligned}
$$

The production for identifiers has been omitted. It is a typical lexer 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.

The semantics of M3L statements will be discussed in the subsequent sections.

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.

A. Concept Definitions and References

A M3L definition consists of a series of *definitions* ($\langle def \rangle$ 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 the 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 visibility of 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 used base concept(s) leads to an error.

Apart from the definition of new concepts the above expressions can be used to augment a concept definition if the leading identifier already has been introduced. E.g., the following expressions lead to the same definition of the concept `NewConcept` as the above variant:

```
NewConcept;
NewConcept is an ExistingConcept;
NewerConcept 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 }
PeterTheMusician is a Peter, a Musician {
  Oboe is a playedInstrument }
```

In this example, we assume that concepts `String` and `Number` are already defined. 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 `PeterTheMusician`. Contexts with those names may be defined later on, though.

Tying a context to a concept can be interpreted in different ways. This is elaborated in Section IV.

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 *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.

 ISBN: 978-1-61208-157-1

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 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`, and 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 }
```

IV. Content Modeling with the M3L

In this paper the M3L is used for concept-oriented content management as motivated in Section II. This section discusses content modeling aspects and their formulation using the M3L.

A. Content Modeling

The goal of the application of M3L for content management is to define content in a context, where both content and context consist of an identifier and nested sub contents or contexts. This leads to the following interpretations of a concept definition `A{B;}`: it may represent ...

- content `A` with partial content `B`,
- content `B` in a context `A`, or
- a context `A` with a subcontext `B`.

Object-orientation has difficulties modeling class and instance variants (Section II-A). Figure 2 illustrates the distinct layers found in object-oriented systems. Objects are instances of classes. Classes can in turn be viewed as instances of metaclasses. The layer of metaclasses is closed; metaclasses can be modeled as instances of metaclasses. In model management systems there may additionally be a meta meta layer. Furthermore, there are relationships between instances (association, aggregation) and between classes (specialization, generalization).

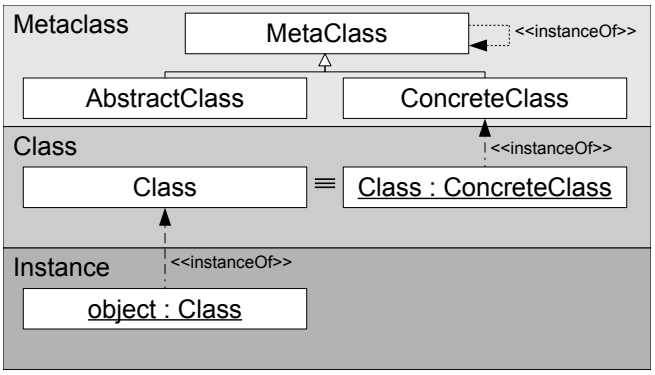

Figure 2. Levels of object-oriented models.

M3L abstracts from these distinct layers and allows interpreting concepts as classes, instances, or variants of instances. An expression `A is a B` can be interpreted in the following ways:

- `A` is a type, `B` is its super type,
- `A` is an instance of type `B`, or
- `A` is a variant, role, or similar of instance `B`.

Expressions with one of the above interpretations are used for various purposes:

- to structure content, e.g.:

  ```
  Picture is a Content {
    imgFile is a ByteArray ;
    title is a String }
  ```

  Here, `Picture` is a content class that defines two attributes `imgFile` and `title`, each of them with "type" information.
- to present content in a context, e.g.:

  ```
  PoliticalIconography is a Context {
    bonaparteCrossesTheAlps is a Picture }
  ```

  A concrete `Picture` is put in a context that helps, e.g., interpreting it.
- to define a context hierarchy, e.g.:

  ```
  PoliticalIconography is a Context {
    StrengthSymbols is a Context }
  ```

  The area of Political Iconography is defined as a context with the domain of symbols of strength as one of the specific contexts it includes.

These basic modeling means are used for various aspects of content models, some of which are discussed in the subsequent sections.

B. Domain Reuse

Contexts are often defined by a combination of other contexts. This way, existing models may (eventually partially) be reused. Reuse is achieved by putting concepts into new contexts. By the additional contextualization content can receive a new meaning or contribute to an additional use case. As part of the combination of contexts, reused concepts may be adapted for new contexts.

 ISBN: 978-1-61208-157-1

For the following example assume classes `Arts` and `History` to be defined that represent the respective research disciplines.

```
ArtHistory {
  Artist is the Artist from Arts;
  Painter is the Painter from Arts;
  Sculptor is the Sculptor from Arts;
  Epoch is the Epoch from History;
  Artwork {
    title is a String;
    artist is an Artist;
    epoch is an Epoch }
  Painting is an Artwork {
    artist is a Painter }
  Statue is an Artwork {
    artist is a Sculptor } }
```

The example shows the definition of a new context for art history that is composed of concepts from the domains of arts and history. The idiom *C* `is the` *C* `from` *M* makes concepts available in the newly defined context: the keyword `from` addresses a source context *M* from which to import a concept *C*. A new concept is defined in the current context as the only refinement of the original concept, but does not add any structure. This way it effectively provides a copy of the original concept from the source context.

C. Variants

Concept definitions in M3L provide a direct means to define variants of a concept in different contexts. Variants of a concept in one context are defined by means of refinement.

The following code shows a quite simple example:

```
Peter is a Person;
PeterTheEmployee is a Peter, a Programmer;
PeterTheHobbyMusician is a Peter, a Musician {
  Oboe is the playedInstrument }
```

We model a `Person` named `Peter`. Specific information on Peter is given in a context-specific way. E.g., the fact that Peter works as a programmer is stated by a concept describing Peter as an employee. A different aspect of Peter's life are his hobbies: in the example he is a `Musician` who—given that `Musician` defines a concept `playedInstrument`—is stated to play the oboe.

D. Revisions

In some applications concepts exist in different *revisions*. Revisions and their relationship have different meanings. Typical content management systems record revisions to reflect the process of content creation. This is often required for legal reasons that demand content states to be reproducible.

When modeling real-world entities through content there is an additional need for revisions. Real-world entities develop over time, so that representations of content may in fact have to cover the process of invention, creation, use, etc. This is particularly true for entities considered in history and art history. In order to reflect a process that typically manifests itself in states of the entity under consideration there have to be concepts for those states. The concepts are refinements of one common concept.

The following example shows two states of the famous painting "Napoleon Crossing the Alps":

```
NapoleonCrossingTheAlps is a Painting;
HistoricalContext {
 NapoleonCrossingTheAlpsSymbol
  is a NapoleonCrossingTheAlps, a Strength; }
Museum {
 NapoleonCrossingTheAlpsArtwork
  is a NapoleonCrossingTheAlps, a Classic; }
```

There is a state where the `Painting` is/was used as a political instrument to visualize strength. Another state represents the painting as a piece of art that is used because of its famousness (independent of the original intention).

E. Personalization

Content management users are provided with predefined concepts that they typically want to tailor to their needs by means of personalization. We investigated the use of personalization for research, teaching, and software engineering.

For example, a model definition like that for Political Iconography above is typically provided as a standardized model. Though standardization is of importance for the cooperation within a domain it is often too restrictive for individual work. Therefore, researchers want to personalize given models both in structure and in content [6].

The following shows an example for the Political Iconography model as shown above:

```
MyPI is a PoliticalIconography {
  Architect
    is the Architect from Architecture;
  Building is an Artwork {
    artist is an Architect } }
```

Here some individual researcher decided to not only consider paintings and sculptures to be pieces of art with political relevance, but to also considers buildings etc. To this end, a new context is declared and the relevant concept is imported.

Inner concepts can be personalized by recursively applying the shown refinement. This way, both sub contexts and aggregated content can be personalized.

F. Content Clusters

Classification of content is used as a parameter to many operations on content. Classification leads to (or is derived from) a clustering of the set of content objects with respect to some notion of content semantics.

Clusters define contexts in which content can be interpreted, delivered, queried, etc. To this end, the notion of context typically incorporates not just one concept at a time, but considers complex contexts for scenarios characterized by multiple aspects of content.

As an example, consider a structure like the following:

 ISBN: 978-1-61208-157-1

```
Tag;
someTag is a Tag;
Content { ... }
someContent is a Content, a someTag;
```

A type `Content` is used to create content objects of a certain structure. These objects can be classified by assigning tags to them. Using M3L's principles, tags are created as `Tag` objects that are used as additional base concepts for content objects.

M3L's production process allows deriving context information based on content. As an example consider the following code:

```
Location { lat is an Int; lon is an Int }
geoLocationOfHamburgPort is a Location {...}
Event {
  date is a Timestamp }
LocalEvent is an Event {
  Location location }
822ndAnniversaryOfPortOfHamburg
  is a LocalEvent
{
  geoLocationOfHamburgPort is the location;
  05/06/2011 is a date;
  05/07/2011 is a date;
  05/08/2011 is a date }
```

In this example we define basic categories for spatial and temporal classification. `LocalEvent` is defined as the base concept for content that carries characteristics from both these classification domains.

New content can be created—independent of these definitions—like this:

```
myPhoto is a Photo {
  ... is the imgData;
  geoLocationOfPortOfHamburg is the location;
  05/07/2011 is the date }
```

Since `myPhoto` fulfills all requirements of a `822ndAnniversaryOfPortOfHamburg`, myPhoto is classified accordingly.

The determination of context from content can be used for a range of operations on content. In particular, there are several uses of pattern-based analysis in community-based content collections like support for queries, tag suggestions for newly added content, and summaries generated for large collections [5].

An increasing number of applications aims to guide users by presenting recommendations for content that is supposed to be of interest to the user. Recommendations heavily depend on context, typically including the tracked users and the user for whom to compute recommendation. The model of a user context may, for instance, consist of an explicit user profile and implicit user tracking information. Recommendations can be defined as personalized content by composition of existing content [2]. In M3L such recommendations can be deduced by applying production rules.

V. Conclusion

Content modeling is of practical importance, both for digital content as well as for content-based descriptions of real-world entities. New applications demand for extended modeling support. Recently, context information was identified as a key ingredient to powerful content models.

Existing content modeling approaches can cover most content modeling requirements, but they are not well integrated. Some model aspects, in particular variants of instances with dynamic classification in changing contexts can only be expressed by auxiliary constructs in many cases.

The Minimal Meta Modeling Language as an abstract modeling language (M3L) is a proposal to an integrated content and context modeling approach. The expressive power of the M3L supports content management tasks in an integrated way, providing modeling constructs for application-specific models.

Acknowledgment

The author would like to thank Joachim W. Schmidt and Sebastian Bossung for fruitful discussions about content modeling. Numerous project partners who participated in research projects on the topic are acknowledged. The author owes a debt to his employer, the T-Systems Multimedia Solutions GmbH, for supporting the scientific work that abstracts and extends the insights gained in practical projects.

References

[1] From content to context to engagement – the future of web content management is context. Executive focus white paper, Ektron, Inc., 2010.

[2] Zeina Chedrawy and Syed Sibte Raza Abidi. An adaptive personalized recommendation strategy featuring context sensitive content adaptation. In *Proceedings Adaptive Hypermedia and Adaptive Web-Based Systems, 4th International Conference*, pages 61–70. Springer, 2006.

[3] Neil Clemmons. If content is king…context is the kingdom…. http://experiencematters.criticalmass.com/2007/12/07/if-content-is-king-context-is-the-kingdom/, December 2007. last visited May 26 2011.

[4] Joseph Goguen. An introduction to algebraic semiotics, with application to user interface design. In *Computation for Metaphors, Analogy, and Agents*, pages 242–291. Springer, 1999.

[5] Lyndon Kennedy, Mor Naaman, Shane Ahern, Rahul Nair, and Tye Rattenbury. How flickr helps us make sense of the world: Context and content in community-contributed media collections. In *Proceedings of the 15th International Conference on Multimedia (MM2007)*, pages 631–640. ACM, 2007.

[6] Hans-Werner Sehring and Joachim W. Schmidt. Beyond databases: An asset language for conceptual content management. In *Proceedings of the 8th East European Conference on Advances in Databases and Information Systems, ADBIS 2004*, pages 99–112. Springer-Verlag, 2004.

 ISBN: 978-1-61208-157-1