Content Management: An Area of Research and a Melting Pot of Approaches from Neighboring Fields.
28 Sep 2011, CONTENT 2011, Rome, Italy

Dr. Hans-Werner Sehring, T-Systems Multimedia Solutions GmbH, Germany
Contents of this Talk.

- Introduction: Content Management state-of-the-art.
- Advanced Content Management Tasks and Requirements.
- Concept-oriented Content Management (CCM).
- Content Modeling.
- CCM System Architecture.
- CCM System Generation.
- Content Visualization.
- Summary.
INTRODUCTION: CONTENT MANAGEMENT STATE-OF-THE-ART.
Introduction.

Content Management: an Industrial Branch, not a Research Topic.

- The analysis of the area of **Content Management** and the development of **Content Management Systems (CMSs)** is industry-driven.
- Contributions for CMSs are spread over a **range of areas**, including but not limited to:
  - Information systems and databases.
  - Search and retrieval, automatic classification and clustering.
  - Web-based information systems.
  - Semantic web.
  - E-business.
  - Workflow systems.
  - Authoring tools.
- Content management as an area of research has **not matured enough** in order to come up with clear definitions, principles, methodologies, models, patterns, ...
- ⇒ Need for this CONTENT conference.
Introduction.
Content Management.

- There is no agreed-upon notion of content management.
  - Web content management.
  - Enterprise content management.
- Perhaps not even of content.
- We consider a “many-of-a-kind” scenario here.
  - Structured content.
  - Services over uniform content.
  - Automatic, template-based rendering.
- We do not consider as content management:
  - Singular documents.
  - Unstructured content.
  - Wikis etc.
Introduction.
Basic Content Management Requirements.

- **Content Creation:**
  - Import and syndication of content.
  - Authoring and modification of content.
  - “Web 1.0”: by dedicated editors; “Web 2.0”: by every user (*user-generated content*).

- **Quality Assurance:**
  - Domain-specific, e.g., following business rules. Legal aspect ➔ dedicated editors.
  - Technically, e.g., no “broken links”.

- **Distribution:**
  - Rendering of content into documents.
  - Delivery / playout of multimedia documents, e.g., HTML pages.
  - Export of “raw” content.

- Inclusion of **additional services** in delivery:
  query, retrieval, classification, recommendation, e-commerce, tracking, personalization, ...
Introduction.
So, What is Content?

- Well known: hierarchy(?) of
  - Data.
  - Information.
  - Knowledge.

- Content?
  - Technical definition.
  - Pragmatic definition.
  - Epistemic definition.
Introduction.
Technical View: from Multimedia Documents to Content.

- An observation: in documents content, structure, and layout can be separated.

A laid outed document:

1. Introduction
Content management systems *have become*

Its content:

Introduction
Content management systems *have become*

Its structure:

- section
- paragraph
- emphasis

- or -

(style is content?)

Introduction
Content management systems *have become*
Introduction.
Pragmatic View on Content.

- If we see content management as the discipline of **database publishing**: content is data with the purpose of being ...
  - Created in an editorial process.
  - Quality assured.
  - Rendered into a **document**.
  - Published / played out / sold.
  - Syndicated.
  - Perceived.
  - ...
Introduction.
Epistemic View on Content.

- Inspired by Ernst Cassirer:
  content is the **meaningful** and **purposeful** representation of entities.

- **Meaningful**: symbol (semiotics: sign) composed of ...
  - Something perceivable.
  - Some concept for a class of (important) perceptions.

- These two are indivisible aspects of content
  (see critics on, e.g., Kant’s categories; Ontology vs. Epistemology in philosophy).

- ☐️ not just “typed data”.

- **Purposeful**: depending on some context.
  - Perceiver’s knowledge.
  - Perceiver’s task.
  - Perceiver’s location.
  - ...

---

Content is a knowledge structure, concepts as well as evidence.
Introduction.
Content Management Models.

- In addition to the above models
  - **Content model** (structure, schema).
  - **Layout model**.

A range of **further models** is required for a complete presence (site, hypermedia, ...):
  - **Navigation model**.
  - Browser / device **capability description**.
  - **User model**.
  - ...

- When models are missing, they are often hardwired into other models, e.g.,
  - Navigation in layout.
  - Capabilities in layout.
Introduction.  
Contemporary Content Management Systems.

- The relevance of content management has brought up a plethora of content management systems (CMSs).

- **Contemporary CMS**: built as information systems following 3-tier architecture.
  - Data layer.
  - Application layer.
  - Presentation layer.

- Content is data in databases, playout is “database publishing”.

- Thus implemented as *generic software*.
  - Parameterizable with content structure and templates.
  - Need to carry conceptual model over all layers and into the users perspective.
  - Need for user to interpret data as content.

- There are products with different properties, but basically all are built along these lines.
Introduction.

“Conventional” CMS Architecture Diagram.
Introduction.
A Critique on Contemporary Content Management Systems.

- Contemporary CMSs deliver **pragmatic solutions**, but some problems still **call for more elaborated solutions**.
- **A-priori conception**, separating creation from operation phase.
  - **Software**: no co-evolution with content, built upon solid conception
    ⇒ not able to evolve with new conceptions.
  - **Content**: does hardly survive schema changes (migration!), does not cross organizational borders.
  - **Schema**: introduces standardization (like in DB, IS) within one organization
    ⇒ content syndication means copying content ⇒ no “real” cooperation.
- **Insufficient multi-channel / x-media support**: content is not really reusable on different media (shortening text, simplifying navigation, adapting services, ...).
- **No modality** (consideration of user: preferences, task, environment, ...).
- **Not suited for Web 2.0 applications** (typical problem of commercial CMSs: optimized for massive playout, no “feedback” channel).
Introduction.
Shortcomings of Contemporary Content Management Systems.

- **Advanced CMS properties:**
  - Models and software evolving with content.
  - Models and software adapting to different modes (of user, devices, ...).
  - Consideration for the different life cycles of content and software.
  - Interrelationship between content model and domain / business model.

- **Two CMS application examples:**
  - **research support system**
  - **(active) learning support system**
ADVANCED CONTENT MANAGEMENT TASKS AND REQUIREMENTS.
Content Management Tasks and Requirements. Content / Knowledge Management.

- **Entity descriptions:**
  - *Data* often describe real world entities.
  - Claim: the same holds for multimedia *content.*

- **Observation** (see, e.g., Cassirer):
  - Natural sciences describe *things* in *states.*
  - Humanities look at *document-based processes.*

- **Entity representation** (see, e.g., Peirce):
  - Representation of entities by *multimedia content* and
  - Interpretation of content by means of a *conceptual model.*
Content Management Tasks and Requirements.

Content Descriptions: Example.

- **Emperor**
  - Type: Equestrian Image
  - Size: 100cm x 50cm
  - Medium: Oil on canvas

- **Regent**
  - Type: Equestrian Image
  - Size: 100cm x 50cm
  - Medium: Oil on canvas

- **Hannibal**
  - Type: Equestrian Image
  - Size: 100cm x 50cm
  - Medium: Oil on canvas

- **Strength**
  - Meta / description data
  - Web 2.0, Collaborative Tagging
  - Semantic Web, ontologies; folksonomies
  - Epistemic structures

- **Equestrian**
  - sits on

- **Horse**
  - represented as

Int. Conf. on Creative Content Technologie 2011 Dr. H.-W. Sehring / Content Management Keynote 28.09.2011
Content Management Tasks and Requirements.
User Adaptivity: Personalization

„group knowledge“

Strength
Equestrian
Horse
Regent
Emperor
Hannibal

user’s view

Equestrian Image

user’s view

Adaptive Visualization, CONTENT 2009, Dr. Hans-Werner Sehring

November 19th, 2009
Content Management Tasks and Requirements.
User Adaptivity: Communication.

Personalization may require model changes.
Communication requires common model.

⇒ A dilemma?
CONCEPT-ORIENTED CONTENT MGMT.

Many aspects joint work with Joachim W. Schmidt. Some aspects joint work with Sebastian Boßung.
Concept-oriented Content Management (CCM). Openness and Dynamics of Modeling.

- **Requirements:**
  - Users want to adapt *(personalize)* content to their current working context individually, while
  - existing content is preserved and
  - cooperation between users is maintained.

- **Demand:** *open* and *dynamic* environments
  - **Openness:** models ...
    - are not limited to predefined concepts and
    - can be changed at any time.
  - **Dynamics:** content management systems ...
    - follow model change without interrupting the domain experts’ work and
    - maintain existing contents and communication structures.
Introduction.
Example for More Demanding CMSs: Orthogonal Personalization.

- **Content personalization:**
  - Values and bindings (content, concepts).
  - Includes presentation personalization.
- **Structure personalization:**
  - Schema (attributes, relationships, constraints).
  - Categories (create new, recategorize).
- Implemented through openness and dynamics.
Concept-oriented Content Management (CCM). Contributions.

- The key requirements of openness and dynamics are met by the **Concept-oriented Content Management (CCM)** approach by means of its key contributions:
  - **Modeling language**
    - Modeling performed by domain expert.
    - Open for changes.
  - **Model-driven system development**
    - Incremental generation.
    - Fully automatic, without developer intervention.
  - **Software architecture**
    - Component-based.
    - Evolution friendly.
CONTENT MODELING.
Content Modeling.
Structuring Content.

- CMS services typically require a **content model**.
- In conventional CMSs: proprietary content schemata (mostly = data schema).
- **Requirement:** 1, $1, or 1€ depending on ...
  - Application layer (e.g., computation).
  - Presentation layer (e.g., rendering the currency sign).
- **Better:** rich central model from which all layers are deduced.
  - While providing openness.
  - While providing dynamics.
- **Compare:** domain modeling.
Content Modeling.
CCM Modeling Language.

- **Definition language** for classes and instances:

```java
model PoliticalIconography
from Regents import Regent
class EquestrianImage refines RegentImage {
  content image : Image
  concept
    characteristic creation : java.util.Calendar
    relationship regents : Regent*
  constraint hasHorse icons >= {horse}
}
class EquestrianImage
let bonaparteCrossesAlps := create EquestrianImage {
  regents := {Napoleon} ...
}
```

- **Query- and manipulation language**: instance lookup, creation, manipulation, and deletion.
Combinations of models for base domains, models as revisions, and personalized variants.

```java
model Political_Iconography
from Regents import Regent
from Artists import Artist
class RegentImage {
    content image :Image
    concept
        characteristic title :java.lang.String
        relationship regents :Regent*
        relationship artist :Artist }

model My_Political_Iconography
from Political_Iconography
import RegentImage
from Humanities import Epoch
class RegentImage {
    concept
        relationship epoch :Epoch
        constraint eqEpoch epoch=artist.epoch }

model Regents
class Regent
class Monarch refines Regent
class King refines Monarch ...

model Artists
from Humanities import Epoch
class Artist {
    concept
        relationship epoch :Epoch
}
class Painter refines Artist ...
```
Content Modeling. The Minimalistic Meta Modeling Language (M3L).

- The CCM Modeling Language is implemented and has been used in various projects.
- A new approach is to use the **Minimalistic Meta Modeling Language (M3L)** for open dynamic content modeling.
- M3L ...
  - Was originally designed for software engineering purposes, in particular model-driven development.
    - Model validation.
    - Model-to-model transformations.
    - Model-to-code transformations.
- Presented for the first time on CONTENT 2011: **M3L for content modeling**.
- M3L addresses several issues more directly than the CCM Modeling Language, in particular contexts and modality.

- **Bootstrapping**: CCM requires an initial model as a starting point for the open dynamic modeling process.
- **Required**: systematic support for domain experts in finding suitable models.
  - Usually, some **modeling expert** (analyst) is consulted.
  - Due to dynamics requirement, such a modeling expert **cannot be employed** in CCM.
  - Domain experts are not modeling experts.
  - But: experts can “tell their story” by **providing examples**.
- **Start with content expressions**:
  - Content abstractions and applications: assigned **names** and bound **values**.
  - Semantic types (concepts): no inner structure.
- Free-form entity descriptions are used as samples; later they become instances of classes.

Joint work with Joachim W. Schmidt, Sebastian Boßung, and Henner Carl.
Content Modeling. Agile CCMS Development.

- **Agility:**
  - Modeling process based on the possibility to generate CCMSs dynamically.
  - Domain experts review their models based on experiences with an operational CCMS.
  - If changes to the model are required, another iteration of the process is started.
  - Entity descriptions created within the CCMS can be used as samples for the next iteration of the process.

Create content instances.

Construct schema.

Generate CCMS.
CCMS ARCHITECTURE.
CCMS Architecture.
Modules and Components.

- Software and content need to be decoupled in the lifecycle of a CMS.
- In CCMSs, this is achieved by separating two levels of reuse:
  - Content.
  - Code.
- The architecture reflects this in its two main building blocks:
  - Components.
  - Modules.
- Components allow content reuse: varying functionality over same code base.
- Modules allow code reuse: base functionality that can be combined in various ways.
CCMS Architecture.
Components.

- ADL model relationships support a range of operations. The two most commonly used ones,
  - Domain reuse.
  - Personalization.
- map to two specific ways of component combinations:
  - Application structure.
  - Organizational structure.
- Components cooperate along these two dimensions.
- Other model relationships are addressed by one of these architectural means, e.g.,
  - Revisions in the organization structure.
  - Rights Management in the cooperation structure.
A CCMS consists of components.

- One component for each domain model.
- Cooperation of domains for openness.
- Personalization of domains for dynamics.
CCMS Architecture.
CCM Fine-grained Architecture: Modules.

- Modules are software artifacts generated for a content model at hand.
- The combination of module instances within one component determines the CCMS's behavior.
- Dynamic reconfiguration at runtime for dynamics by:
  - Separation of Concerns through module kinds with certain functionality.
  - Uniform module API.
  - Stateless modules.
Central building block of the CCMS architecture: **Mediator**
- **Wrapper** $\rightarrow$ components for the adaption of instances
- **Mediator** $\rightarrow$ coordination of other components

![Diagram showing the CCMS Architecture]

- **Services for model 2**
- **Coordination of sub systems**
- **Services for model 1**
  - Access to base system 1 according to model 1
- **Access to base system 2 according to model 2**

- **Base system 1**
- **Base system 2**
CCMS GENERATION.
CCMS Generation.
Compiler Framework.

- Modeling openness requires adequate software support, prohibits generic software.
- Modeling dynamics requires dynamic software evolution, prohibits manual intervention.

- A dilemma?

- CCMS architecture enables software evolution.
- Software generation to allow dynamic software evolution.
- Based on ideas from MDSD.
- But: not arbitrary functionality, but software fulfilling standard requirements for CCMSs w.r.t. a particular content model.

- **Solution:** standardized functionality (generated) for individual models.
- Basis for all CCMS: **compiler framework**.
CCMS Generation.
CCM Model Compiler Framework.

- CCM model compilers follow established compiler construction principles.
- **Generalizations** of compiler construction:
  - Factor out commonalities into a **framework** extendable by **recognizers** and **generators**.
  - Access to multiple related models from **repositories / dictionaries** instead of “includes”.
- **Extensions** of compiler concepts:
  - Complex **abstract syntax tree** decoration.
  - Rich **symbol tables**.
  - Coordination of various **generators** (backends) contributing artifacts.
CCMS Generation.
CCM Model Compilation Process.

- Compilation process overview (example):

Domain model (source)

```plaintext
model M
class A refines B {
  content ...
  concept ...
}
class B {
  ...
}
```

Intermediate model (parse tree)

- Parse
- Generate

API

```plaintext
interface AbstractA { ...
}
interface A extends AbstractA { ...

abstract intf. for A AbstractA
persistent intf. for A A
...
factory for B BFactory
```

DDL

```plaintext
create table ATab ( ...
)
```

```plaintext
table for A ATab
table for B BTab
```

Code

```plaintext
class AImpl implements A, ...
{
  ...
    select from ATab...
}
class for A AImpl
class for B BImpl
```
CONTENT VISUALIZATION
Content Visualization.
Open Dynamic Content Presentation.

- Openness requires rich, well-designed visualizations. Dynamics requires adaptable visualization (adaptive visualization not possible because of human preferences/needs).
- Generated UIs do usually not meet users’ needs.

- **A dilemma?**

- ⇒ **Solution: Apply CCM principles to visualization.**
  - Represent UIs by open models.
  - Provide initial UI definitions with domain models.
  - Users can reuse and personalize UI definitions.
  - Software is generated from the definitions dynamically.
Content Visualization. Models for Visualizations.

Components
- GUI Component
  - Container
    - ...

Implementations
- Java Container
  - Swing Container
    - javax.swing.JPanel

Technologies
- Technology
  - Description
    - Java
      - HTML
      - SwiXML

Technologies
- AWT
- SWT
- Swing

Visualization environment

Domain expert

Layout
- ...

Application domain
- WELObject
- PISubject
  - Work
    - Picture
    - Movie

Technology
- Visualization Code

(CCM) Compiler

(T·Systems)
Content Visualization.
Example: Rich Client for a Digital Library.
SUMMARY.
Summary.

- Content Management is a **vital industry**.
- Contemporary content management systems address requirements **in a pragmatic way**.
  - Solving the practical problems successfully.
  - Do not scale well to more recent challenges.
- Advanced content management tasks call for **more elaborated support** that in turn calls for a solid conceptual basis, including work on:
  - Modeling.
  - Architectures.
  - (Model-driven) software engineering.
  - Visualization.
- To this end, content management needs to be recognized as a **field in its own rights**.
- **Concept-oriented Content Management** has proven successful.
- Much, much more work needed.
THANK YOU!