Keynote
Aligning Domain, Content, and Software Modeling in Web-based Information Systems

THE EIGHTH INTERNATIONAL CONFERENCE ON CREATIVE CONTENT TECHNOLOGIES (CONTENT 2016)
ROME, ITALY, 23 MARCH 2016

Creative Content Technologies. Modeling. Keynote.

Since 1995. Namics.

Dr. Hans-Werner Sehring. Senior Solution Architect.

Agenda.

FIRST	**Namics in a nutshell.**
SECOND	**Content Management.**
THIRD	**Modeling Approaches.**
FORTH	**Overall Web Site Modeling.**
FIFTH	**Summary and Outlook.**

FIRST

Namics in a nutshell.

NAMICS IN A NUTSHELL

What we offer.

Jürg Stuker

CEO NAMICS

“We are pioneers and experts in the field of digital transformation – and we have been since 1995. As an independent, interdisciplinary full-service partner, we work with you to digitize your business models and critical processes. Your long-term success is the focus of everything we do.”

Our services.

NAMICS IN A NUTSHELL

The facts.

Since 1995. Close proximity. Namics.

6
LOCATIONS

20
YEARS

SECOND

Content Management.

CONTENT MANAGEMENT.

Overview.

Overview Over Content Management.

Content is …

Managed centrally, …

Available in **variants**, …

Published on **channels**, and …

Rendered in **media**, using …

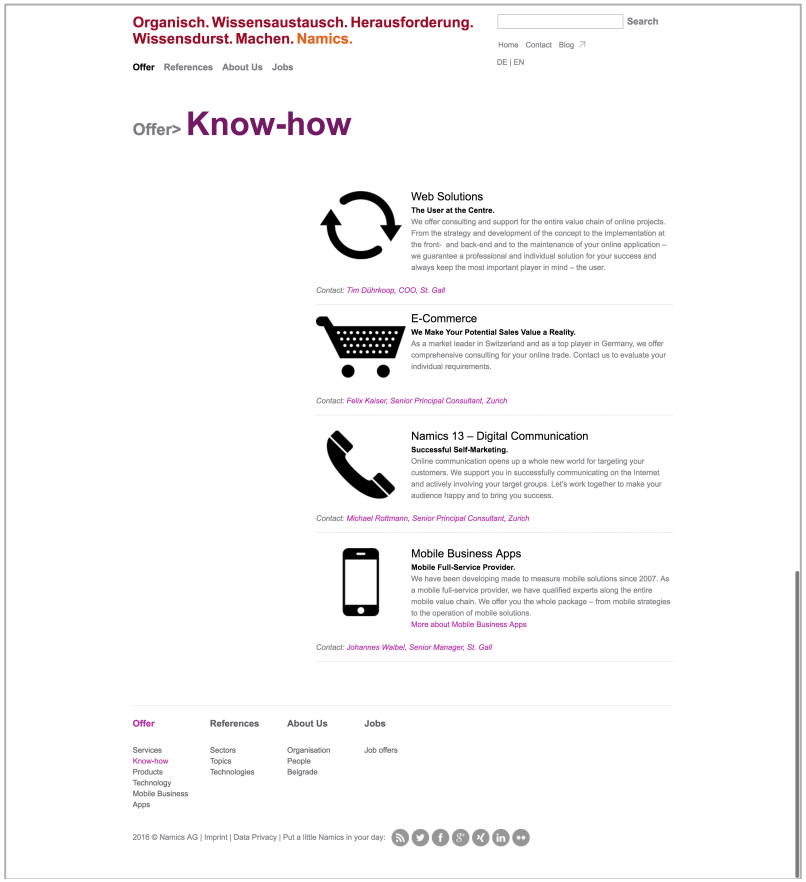
Channel- and media-specific **layouts**.

Typical in CMS applications:

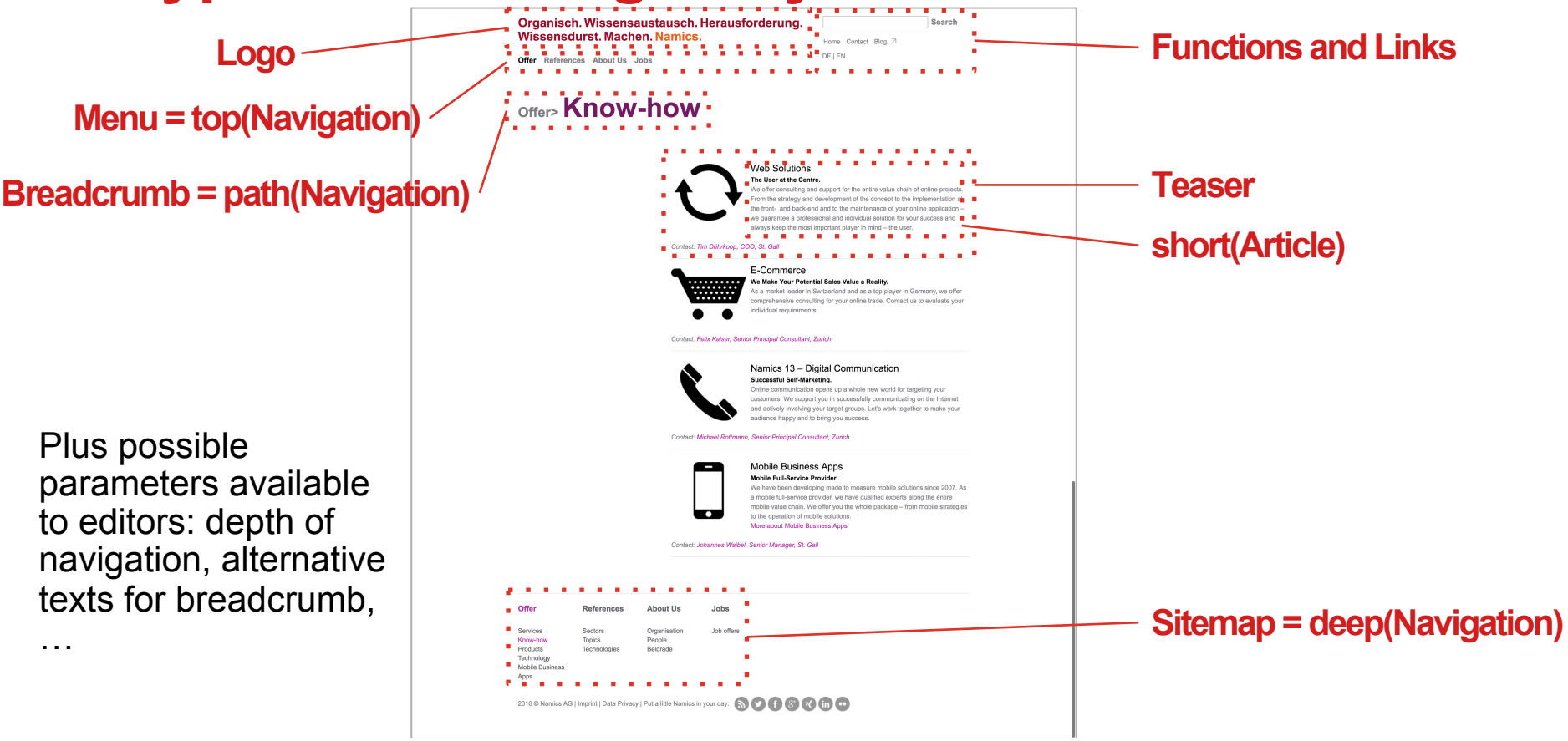
– The choice of layout is performed algorithmic.

– Editors can influence the choice using **layout hints**.

A Typical Web Page Layout.



A Typical Web Page Layout.



Plus possible parameters available to editors: depth of navigation, alternative texts for breadcrumb, …

Considerations for Content Models.

Content management based on a **content model**.

Content model:

– designed for different **recipients**.

– designed for competing **requirements**.

– covering different (conflicting) aspects of content management:

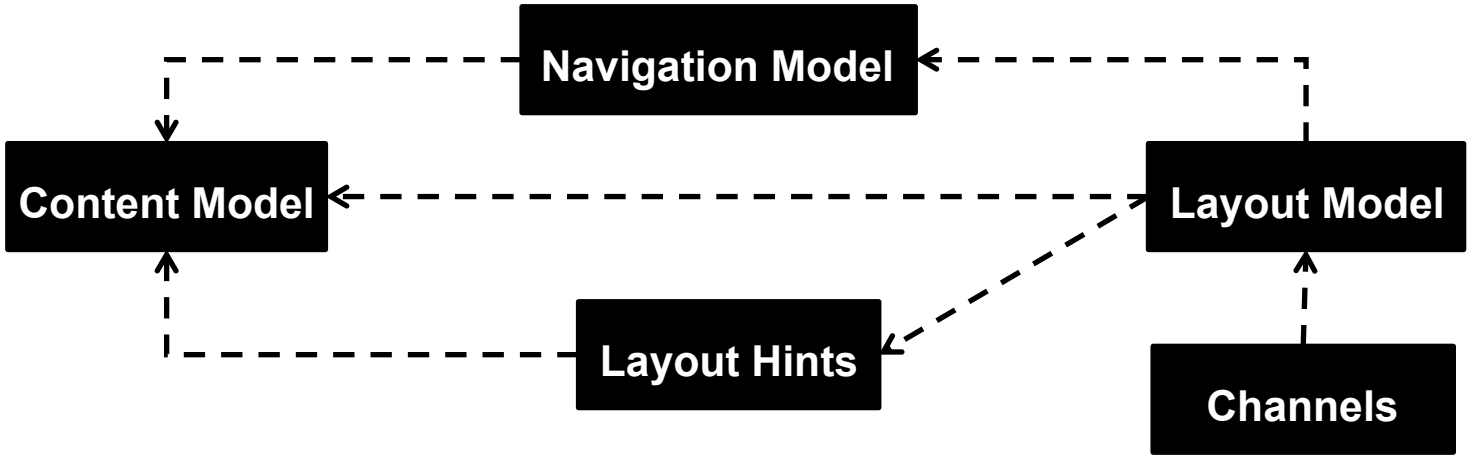
  – **long-lived** content, reflecting **domain model**.

  – **reusable** content, for different **channels**.

  – **variants** of content, e.g., for campaigns and **targeting**.

  – providing **layout-specific** information, document creation.

Separation of Concerns by Distinct Models.



The arrows (borrowed from the UML) indicate dependencies, meaning that instances created from a dependent model refer to instances of the supplier model. In some CMS products the direction of associations is important, since referred content is published together with its referrer for referential integrity. Otherwise, abstractions have to be introduced.

Relating the Separated Models of a Web Site.

Separation of models has been well-known since the works on multimedia representations.

The approaches have somehow been “lost” in the approaches to the world wide web and current content management systems.

Paying attention to the requirements

– to keep models in a **central place** and

– to make them **coherent**,

we investigate approaches to **relate and align** separately reusable models and content.

Previous Work.

Combining different approaches in order to address the duality of long-lived, presentation-independent content and content marketing based on presentation-oriented content optimization.

Concept-oriented Content Management (**CCM**):
– Domain modeling by content and its conceptual model.
– Separating models, connecting and relating those models.

Minimalistic Meta Modeling Language (**M3L**):
– Leveraging distinction between types and instances.
– Modeling content based on concepts in contexts.

Use M3L to separate the modeling concerns using CCM principles, and to recombine them in a flexible way.

CONTENT MANAGEMENT.

On Content Modeling.

Content Modeling Requirements. Current requirements.

Above conventional content management paradigms, additional requirements arise. Content needs…

- **Concepts**.
  - Conceptual model for editing.
  - Semantics for interpretation and rendering.
- **Context**.
  - Multi-channel, X-media publishing.
  - Modality for interpretation and targeting.

Content Modeling Requirements. Current requirements (cont'd).

Need for **evolution** of both concepts and content.

- Reusable content that is more long-lived than presentations, business models, …
- Effort to migrate content to new presentations, supplemented business models, …
- There are domain entities that represent process steps rather than most current states.

Independent evolution of …

- Content,
- Layout, ...

Content Modeling Considerations. Pursuing Multiple Requirements at Once.

Content model recipients: simultaneously designed for …
– editor,
– programmer, and
– system.

Content model utilization: accordingly designed for …
– ease of use,
– performance,
– long-term storage, and content reuse.

Content model focus: simultaneously allowing content to be …
– long-lived, for domain modeling
– layout-specific, for document creation
– layout-agnostic

THIRD.

Modeling Approaches.

Modeling Approaches Overview.

Typical practice: all aspects of content and additional information stored in **central content base**, all included in its content model.

Problems, as laid out before: content that survives relaunches, multi-site management, multi-channel publishing, …

Studies so far: studied domain modeling with **Concept-oriented Content Management** (**CCM**), currently applying **Minimalistic Meta Modeling Language** (**M3L**) to content modeling for **multi-everything content management**.

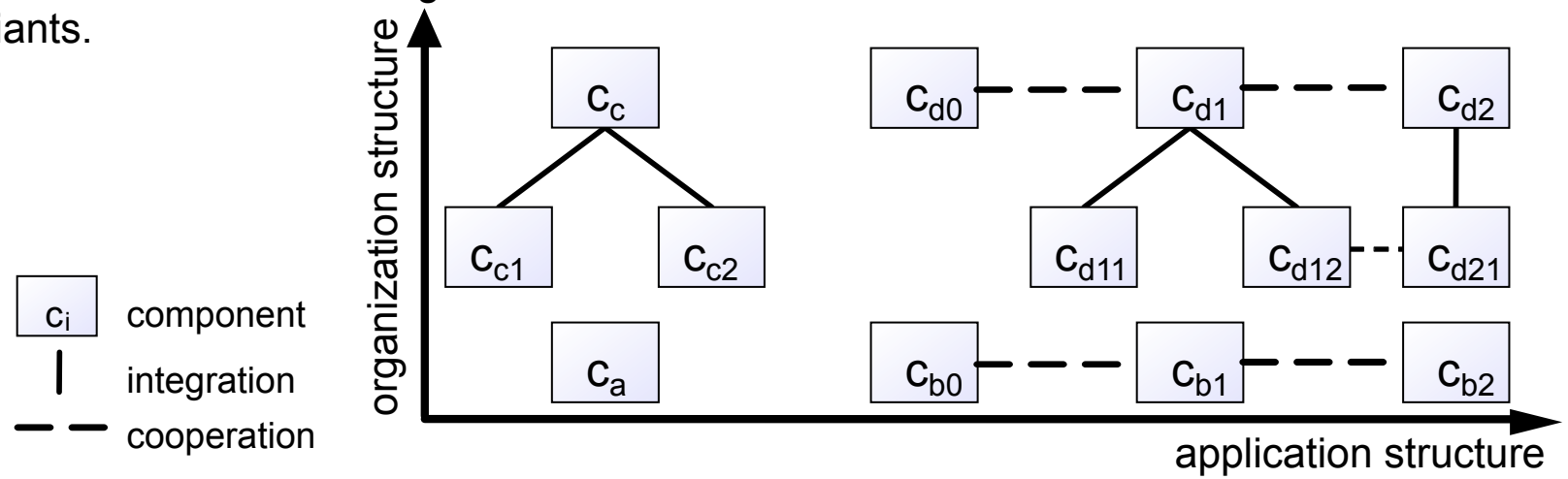
MODELING APPROACHES.

Concept-oriented Content Management.

A Glance on Concept-oriented Content Management (CCM).

Lots has been published on the CCM approach. It has many features that are realized by a certain way of implementation.

For the sake of today's discussion we concentrate on model/ component distribution: separate domain models, cooperation for domain combinations, integration of model revisions and to derive variants.



MODELING APPROACHES.

M3L Foundations.

M3L.
Origins of the Language.

Ongoing work on the Minimalistic Meta Modeling Language (M3L).

Originally designed for SW engineering purposes, model-driven development (MDD) in particular.

Goals:

- Open modeling: concepts are interpreted w.r.t. a context, concepts may be changed.
- No destructive updates, but co-existing variants of objects.
- Evaluations on model level for model checking, production of models and code for MDD.

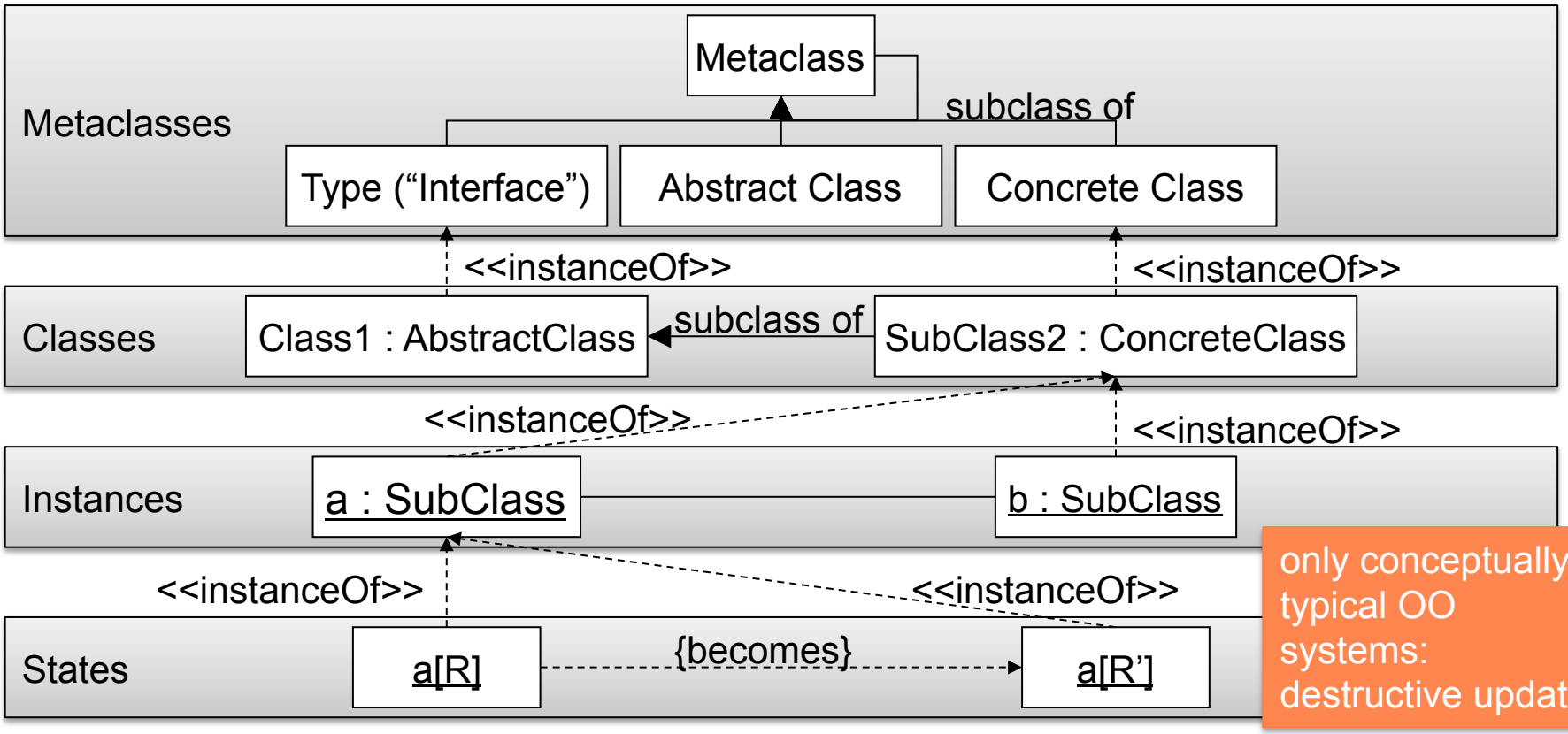
M3L.
Origins of the Language (cont'd).

Shows properties suitable for content modeling.

- Object-oriented properties, but abstraction from object-oriented principles.
- Variants and contexts as the primary idioms.
- Separation of concerns, separating models of different domains / abstraction levels.
  E.g., conceptual classes, concrete classes, …

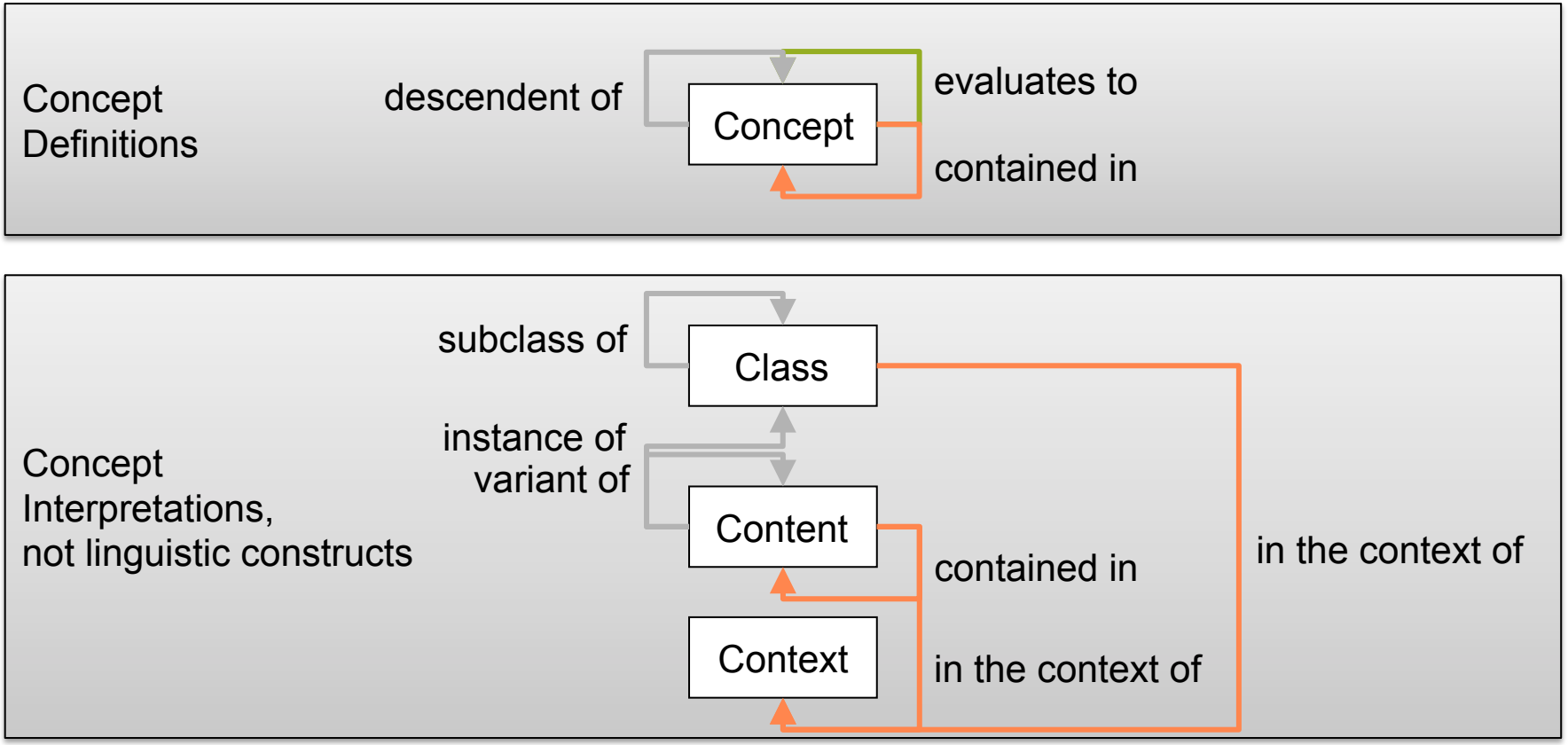
Levels of OO Entities.

here: illustration by UML class / object diagram



only conceptually; typical OO systems: destructive update

Abstraction From Levels of OO Entities.



MODELING APPROACHES.

M3L Definition.

Some M3L syntax. References.

To keep code examples short, we use the short M3L notation without keywords.

Reference to a: a;

Reference to b in context a: a { b; }

set consisting of a and b: { a; b; }

a and b are identifiers defined before.

An identifier is a string that does not contain whitespace. In order to contain whitespace, identifiers my be quoted:
"this concept" "defines this one";

Some M3L syntax. Definitions.

Definition of b as a subconcept of a: a b;

definition of b as a subconcept of a with content d; d is defined in the context of b as a subconcept of c: a b { c d; }

definition of b as the set consisting of c and d: a b { c; d; }

definition of b as a subconcept of a with value c: a b |= c;

definition of b as a subconcept of a with a semantic rule producing a fresh c (named d): a b |= c d;

definition of b as a subconcept of a with a semantic rule producing/ recognizing according to the given template:

a b { c d ; } |- "Hello" d "!";

Some M3L syntax. Redefinitions.

Given a definition a b;.

Redefinition of b adding a further parent concept c: c b;

Redefinition of b adding content d as a subconcept of c:
b { c d; }

Redefinition of b adding content c; if b does not contain c, add it; answer c in the context of b; c has to exist in the context of b:
b { c; }

Redefinition of b with reference to c as semantics: b |= c;

Redefinition of b with c as input/output: b |- c;

Some M3L Pragmatics.
Interpretation of concept relationships.

Possible interpretations – these are not made explicit.

Specialization/generalization, instantiation, states, materialization:

A B; → A is a generalization of B, B specializes A
→ B is an instance of A, A is the type of B
→ A is an instance, B is a state of A, B is a role of A
→ B is a materialization of A

Association, aggregation (class level):

A B { C D; } → instances of B refer to a C under the name D
→ instances of B aggregate C's under the name D
→ class or instance D is defined in context B

Association, aggregation (instance level):

A B { C D; } → B refers to D
→ B aggregates D

Content Example.

```
Object Person { String name; }              # Object, String may given
```
→ definition of Person; definition of name in the context of Person

```
Person Peter { name "Peter Smith"; }
```
→ concrete definition of Peter; definition of Peter Smith in the context of Peter; name is “inherited” (exists in context of Peter)

```
Object Instrument;                          # “type definition”
Instrument Flute;                           # “instance creation”
Person Musician { Instrument playedInstrument; } # “role definition”
Peter PeterTheMusician { playedInstrument Flute; } # “state def.”
```
→ extension/role/variant/... of Peter, adding one content definition

FOURTH.

Overall Web Site Modeling.

OVERALL WEB SITE MODELING.

Applying M3L.

Web Site Modeling Example Using M3L. Content.

A simple content model consisting of “nodes” of different kinds.

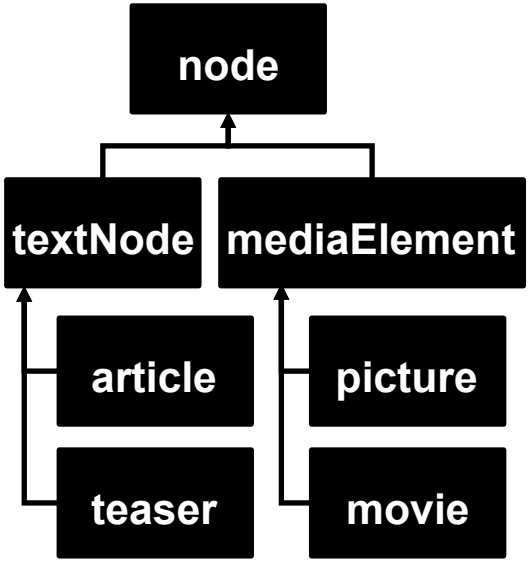
```
Model contents {

  Object node;
  node textNode { String headline; String text; }
  textNode article { String shortHeadline; String shortText; }
  textNode teaser { picture image; node teaseredElement; }
  node mediaElement;
  mediaElement picture;
  mediaElement movie;

}
```

Model, Object, String may be given.

note: no references to layout



Here: Global, (hopefully) stable model.

Web Site Modeling Example Using M3L. Navigation.

Navigation is best separated from content in order to maximize the potential of content reuse. In certain scenarios, though, content might depend on its (navigational) context.

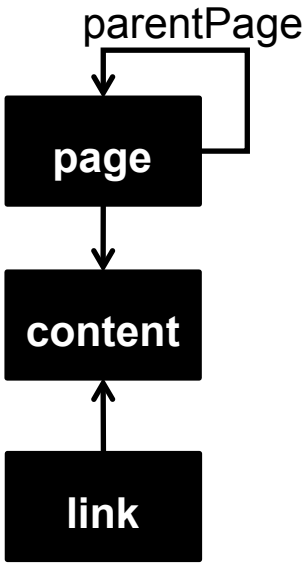
As a simple example, we just introduce (web) pages as navigational entities and links from content nodes to an other content node.

```
Model navigation {

  contents{node;} page { page parentPage; contents{node;} content; }
  contents{node;} link { contents{node;} target; }

}
```

note: no references to specific content



Web Site Modeling Example Using M3L. Layout.

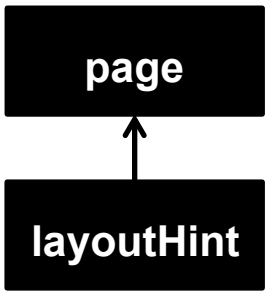
Regardless what the concrete layout is, typically editors need to have (limited) control over the appearance of rendered content.

For the sake of this example, we just introduce a general layout hint.

```
Model layoutHints {

  Object layoutHint {
        navigation{page;} page;
  }

}
```

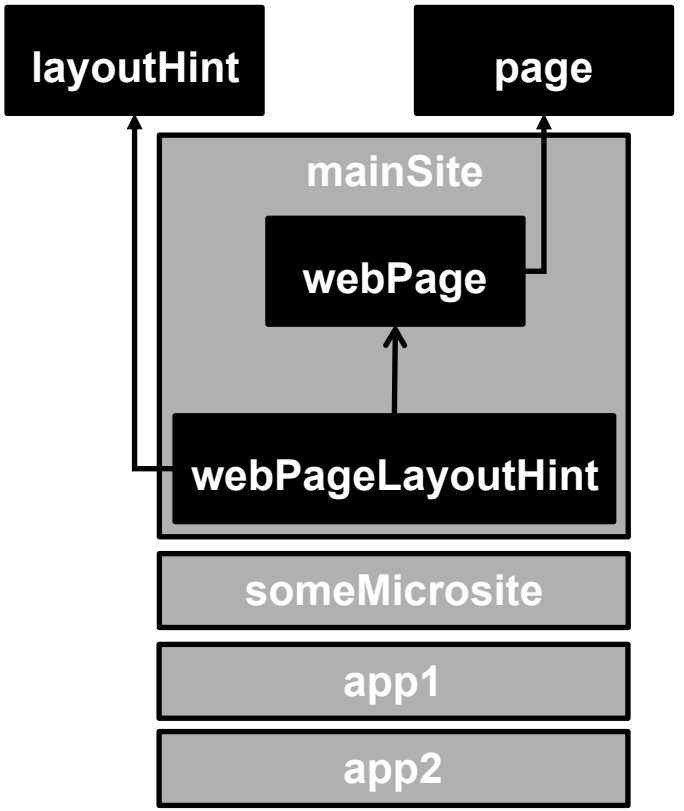


Web Site Modeling Example Using M3L. Channels.

Channels predefine structure and include layout (hints).

```
Model channels {
  channel web {
    channel mainSite {
      navigation{page;} webPage;
      layoutHints{layoutHint;} webPageLayoutHint {
        webPage page;
        boolean hideInNavigation;
      }
    }
    channel someMicrosite;
  }
  channel apps { channel app1; channel app2; }
}
```

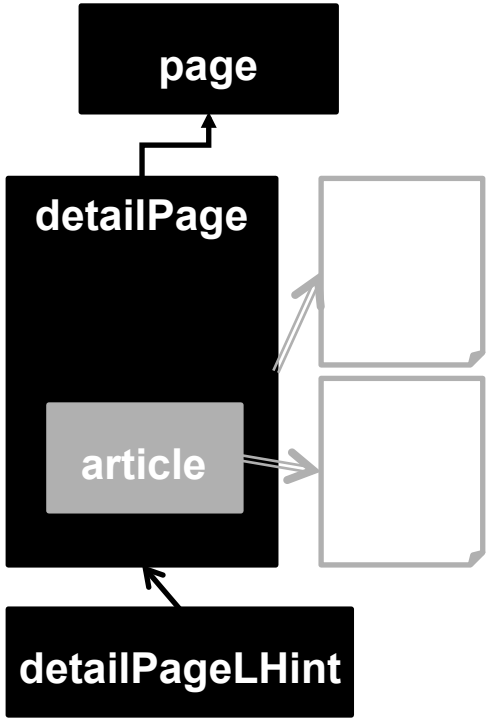
channel may be given.



Web Site Modeling Example Using M3L. Layout.

A layout typically consists of an HTML template plus rules that define how to include content.

```
layout mainSite {
  navigation{page;} detailPage
    |- "<html><head><title>" contents{headline;}
       "</title></head><body>" content "</body></html>"
  channels{web{mainSite{webPageLayoutHint;}}} detailPageLHint {
      page detailPage; hideInNavigation false;
  }
  detailPage {
    contents{article;} |- "<div><h2>" headline "</h2>" text "</div>";
  }
}
```



Web Site Modeling Example Using M3L. A Page Instance.

```
contents myContents {
  article KeynoteRome2016 {
    headline "Separating Models";
    text "In this talk I will present you…";
    picture illustration { … }
  }
}

mainSite {
  detailPage KeynoteRome2016Page {
    parentPage homePage;
    content myContents { KeynoteRome2016; }
  }
}
```

other channels in the same way, using the same content

```
<html>
 <head>
  <title>Separating Models</title>
 </head>
 <body>
 <div>
  <h2>Separating Models</h2>
  In this talk I will present you…
 </div>
</body>
</html>
```

OVERALL WEB SITE MODELING.

M3L Benefits.

Web Site Modeling Using M3L. Context.

M3L allows to manage and use content in context.

Here, one aspect of context is the channel on which content is published.

This way, applying the M3L as shown…

– allows to **reuse content** for different publication channels and

– to **assign channel specific amendments**, e.g., layout hints to content.

Plus, M3L this way provides a rather **compact representation** of content and layout.
Instead of enriching models, put them into context.

Web Site Modeling Example Using M3L. Advanced Modeling.

Particular power of M3L: no distinction between types, content instances, partial instances, content prototypes, …

E.g., prepare “page types” as instances that can be used as **prototypes** by editors.

```
DetailPage ProductPage {

  parentPage productOverviewPage;
  Product content productContent {
    headline |- "Offer of " productName;
  }

}
```

This concept can be used as a blueprint for product pages.

FIFTH.

Summary and Outlook

Summary.

Content management includes more than just content – context, navigation, layout, and control over layout by editors.

Content reuse is limited when all concerns are reflected in the content model. Such content typically needs migration when changing layout. In multi-channel applications the content model becomes overloaded.

Using **distinct models** for separation of the concerns is the obvious remedy to this. Instances created according to the different models need to be **reintegrated**. **Dependencies** between the models must be introduced carefully, as well as their direction.

M3L as a language and the idea behind it is not only well-suited for context-dependent content, but also for the other models.

Outlook.

Combination of CCM and M3L:

– M3L definitions combined from various sources, model reuse.

– Content distributed over (specialized) systems – a CCM property.

Advanced rendering:

– Abstraction from textual code for creating distinct presentations.

– Distribution of presentation code over server and client.

Practical insights yet to be gained:

– Reuse of content across channels.

– Variants of content vs. shared content.

hans-werner.sehring@namics.com. Senior Solution Architect.

Thank you. Namics.