TALLENCE

ON INTEGRATED MODELS FOR COHERENT CONTENT MANAGEMENT AND DOCUMENT DISSEMINATION

The Thirteenth International Conference on Creative Content Technologies
CONTENT 2021

April 18, 2021, to April 22, 2021 – Porto, Portugal – Hans-Werner Sehring
hans-werner.sehring@tallence.com

// Dr. Hans-Werner Sehring

Short Bio

- > Since 2020 at Tallence AG, Hamburg, Germany
- > Professional experience:
  - > 10+ years in research
  - > 10+ years in software development and consulting for digital transformation
  - > Topics:
    - > software architecture, (model-driven) software development
    - > content management, ecommerce, personalization, knowledge representation
    - > user interactions on web and mobile
- > IARIA Fellow since 2012
- > Steering committee member of the CONTENT conference
- > Editor-in-chief of the International Journal on Advances in Intelligent Systems

TALLENCE

Dr. Hans-Werner Sehring
Senior Software Architect

Tallence AG
Neue Gröningerstraße 13
20457 Hamburg

hans-werner.sehring@tallence.com

// Outline of the paper

1. Content management systems
   - > Content is represented as data in various forms throughout a content management system
   - > Software components must consistently implement the underlying content model in order to maintain coherence
2. We us the Minimalistic Meta Modeling Language (M³L) to study content modeling and management
   - > M³L was designed with Model-driven Software Development tasks in mind
   - > M³L proved useful for content modeling

> We study coherent content representations using two examples

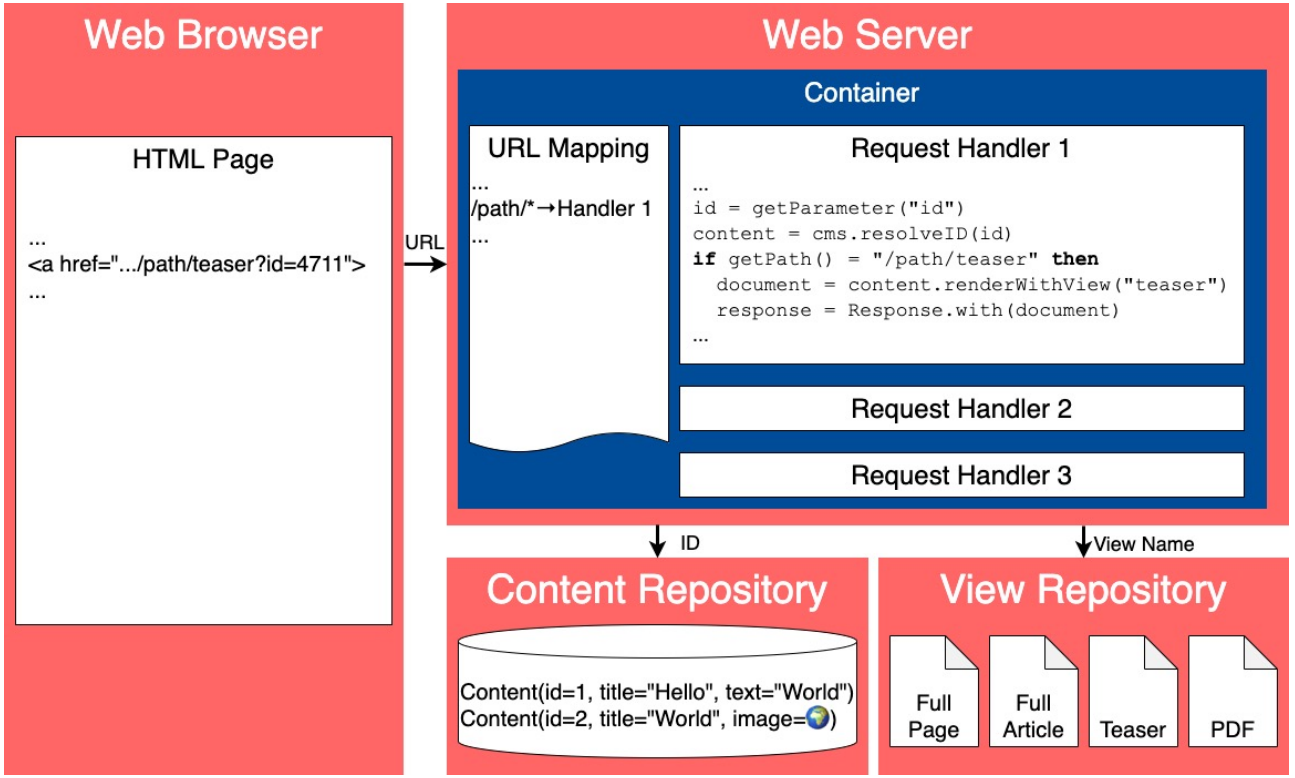
3. Multilingual websites
4. Websites that incorporate campaign management

5. We close with a summary

// Overview over a sample content management application

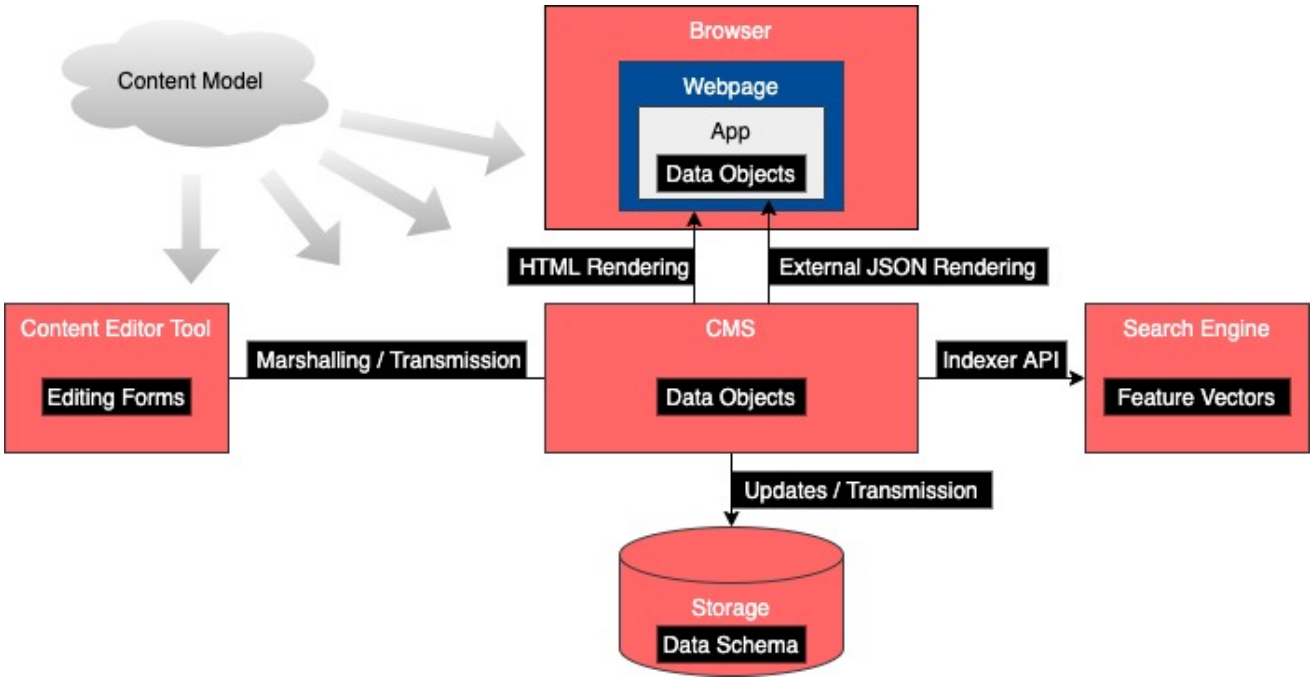
The content and the underlying content model are used in different parts of a content management system

- A content model defines interfaces that are consumed by different audiences, e.g.:
  - An adequate editing tool and support for quality assurance for editors
  - A framework to customize the CMS by application programmers
  - Client-side apps developed by application programmers
  - Templates for layouts by application programmers.



// Encodings of content all over the place

Content management systems consist of various software components that use content representations in various forms



> All schemata, APIs, etc. must conform to the same underlying model.

> In the case of a CMS, this is the content model.

> It shows up in manifold form throughout a system.

// M³L

An overview over (nearly) all language constructs. More complete descriptions can be found in the paper and the literature

`> A`	> The declaration of or reference to a *concept* named **A**
`> A is a B` `A is the B`	> The *refinement* of a concept **B** to a concept **A**; **A** is a specialization of **B**, **B** is a generalization of **A**
`> A is a B { C }`	> Containment of concepts; **C** belongs to the *content* of **A**, **A** is the *context* of **C**
`> A |= D`	> The *semantic rule* of a concept of a concept **A**; whenever **A** is referenced, **D** is bound; if **D** does no exist, it is created in the same context as **A**
`> A |- E F G.`	> The *syntactic rule* of a concept **A**; **A** is printed out as or recognized from the concatenation of the syntactic forms of concepts **E**, **F**, and **G**; if not defined, a concept evaluates to / is recognized from its name

// M³L Expression Evaluation

- The M³L has an operational semantics for expression evaluation
- It is based on (any combinations of)
  - Refinement
  - Semantic rules
  - *Visibility* rules
    - All concepts in the content of a concept are also visible in the content of refinements: `A { B }, C is an A ⇒ C { B }`
    - All concepts in the content of a concept are also visible in the contents of concepts in the context of that concept: `D E { F } ⇒ E { F { D } }`
  - *Narrowing*: If a concept A has a subconcept B, and if all concepts defined in the context of B are equally defined in the context of A, then each occurrence of A is narrowed down to B.

```
Person {
  Name is a String }
PersonMary is a Person {
  Mary is the Name }
PersonPeter is a Person {
  Peter is the Name
  42 is the Age }
```

```
Person {
  Peter is the Name
  42 is the Age }
⇒ PersonPeter
```

```
Person {
  Mary is the Name
  42 is the Age }
⇒ Person {
  Mary is the Name
  42 is the Age }
```

// Basic content management with the M³L

The M³L can be used to express content and content models

> Using the M³L, with a content model like:

```
Article is a Content {
    Title is a String
    Text  is a FormattedString
    Image is an OpaqueContent }
```

(Base types like **Content**, **String**, **FormattedString**, and **OpaqueContent** may be predefined here.)

> According to this mode, e.g., the following content can be created:

```
NewsArticle123 is an Article {
    "Breaking News" is the Title
    "This is a report about ..." is the Text
    Asset456 is an Image
    Asset789 is an Image }
```

// Basic Document Rendering

Also, templating rules for content rendering can be expressed in M³L

> Semantic production rules allow formulating templates for document rendering

> Example HTML rendering:

```
Article |- "<div class=\"article\">"
                "<div class=\"title\">" Title </div>
                "<div class=\"text\">" Text </div>
                "<div class=\"image\">"
                    "<img src=\"" URL from ImageResource {
                        Image is the Content
                    } \"/>
                </div>
            </div> .
```

> (Note that, e.g., ** ** is implicitly defined and syntactically evaluates to its name.)

// Basic HTTP concepts in M³L

Likewise, communication protocols can be defined using the M³L

> Several protocol components are required for web interactions.

> A concept for URLs:

```
URL { Protocol Host Port Path } |- Protocol :// Host : Port Path .
```

> A concept for HTML webpages (simplified):

```
WebPage { Title Content URL } |- <html> <head> <title> Title </title> </head>
                                 <body> Content </body> </html> .
```

> A concept for Cookies:

```
Cookie {
  Name is a String
  Value is a String }
|- Cookie: Name = Value .
```

// Basic HTTP concepts in M³L (cont'd)

> The bidirectional nature of syntactic rules allows to specify inputs and outputs.

> A concept for (HTTP) requests:

```
Request { URL ProtocolVersion Method Parameters HeaderAttributes
          Cookies is a Cookie Body }
|- Method " " Path from URL " " Protocol from URL / ProtocolVersion \n
   HeaderAttributes \n Cookies \n \n
   Body .
```

> A concept for (HTTP) responses:

```
ResourceResponse is a Response { Protocol ProtocolVersion Cookies Content }
|- Protocol / ProtocolVersion " " 200 " " OK \n
   Cookies \n \n
   Content \n .
```

// Basic HTTP concepts in M³L (cont'd)

> A base concept for rules to handle requests by specifying the input/output behavior:

```
WebHandle {
  2 is the ProtocolVersion
  Request { ... }
  |= Response {
    WebPage {
      URL from Request is the URL
    } is the Content
    Cookies from Request is the Cookies
  }
}
```

> In the context of **WebHandle**, e.g., responses are defined for possible requests.

> This concept forms the basis to define web servers.

// Multilingual websites

> Many websites are offered in multiple languages.
The content of such websites is localized. Additionally, the websites content and structure may differ.

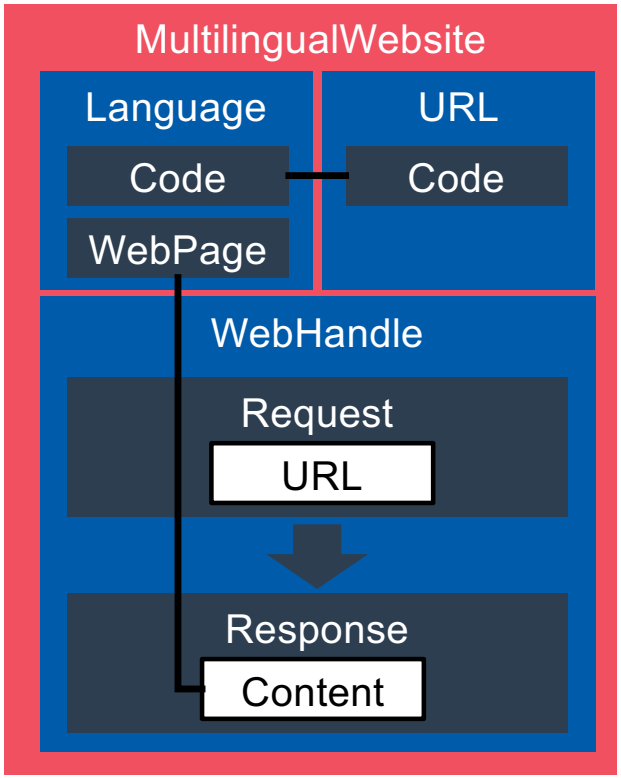
> Typically, users can choose their preferred language.

> As an example, assume a website with multilingual content and webpages that are requested through URLs that have a language code as their first path segment.
E.g., https://www.tallence.com/en/why delivers an English page.

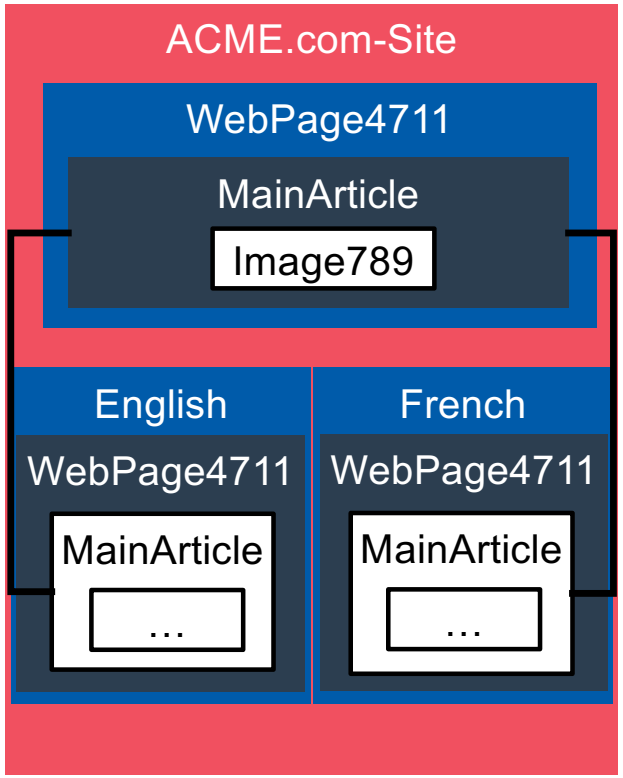
// Basic M³L concepts for multilingual websites

> Concept for the basis of multilingual websites:

```
MultilingualWebsite is a Website {
  Language { Code }
  URL { Code }
  |- Protocol :// Host : Port / Code Path .
  WebHandle {
    Request |= Response {
      WebPage { URL from Request is the URL }
      from Language {
        Code from URL from Request is the Code }
      is the Content }
  }
}
```



// A sample multilingual website

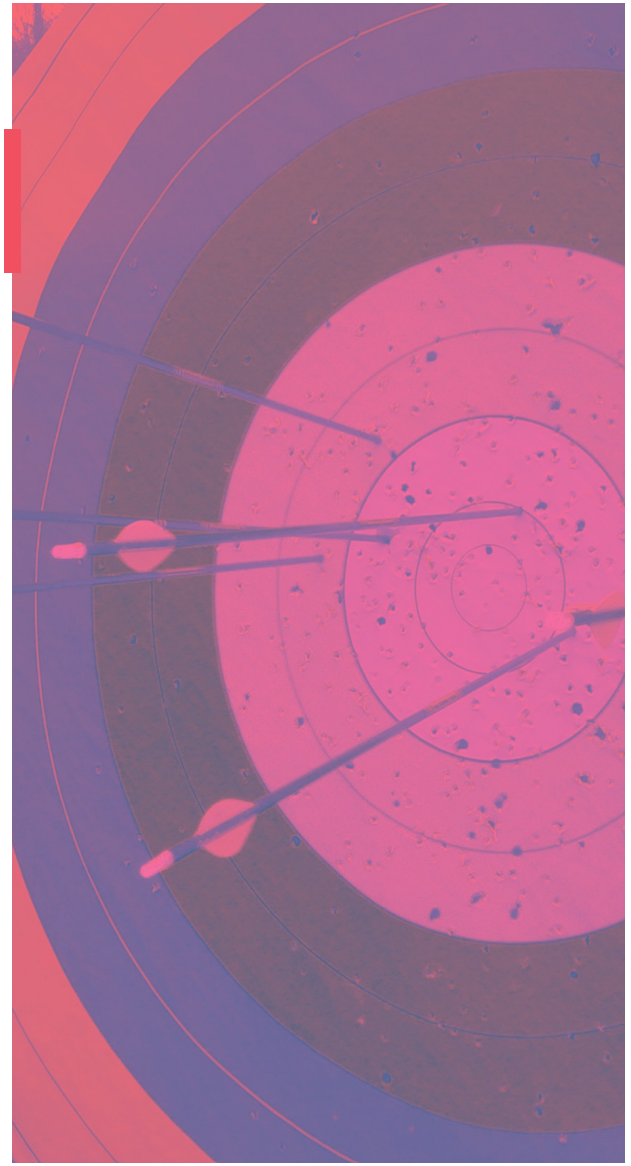


> Code:

```
ACME.com-Site is a MultilingualWebsite {
 WebPage4711 is a WebPage {
  URL4711 is the URL {... /news is the Path ...}
  MainArticle is an Article, the Content {
   Image789 is the Image } }
 English is a Language {
  en is the Code
  WebPage4711 { "Big News" is the Title
                MainArticle {"..." is the Text}}}
 French is a Language {
  fr is the Code
  WebPage4711 { "..." is the Title
                MainArticle {"..." is the Text}}}}
```

// Webpages as part of campaigns

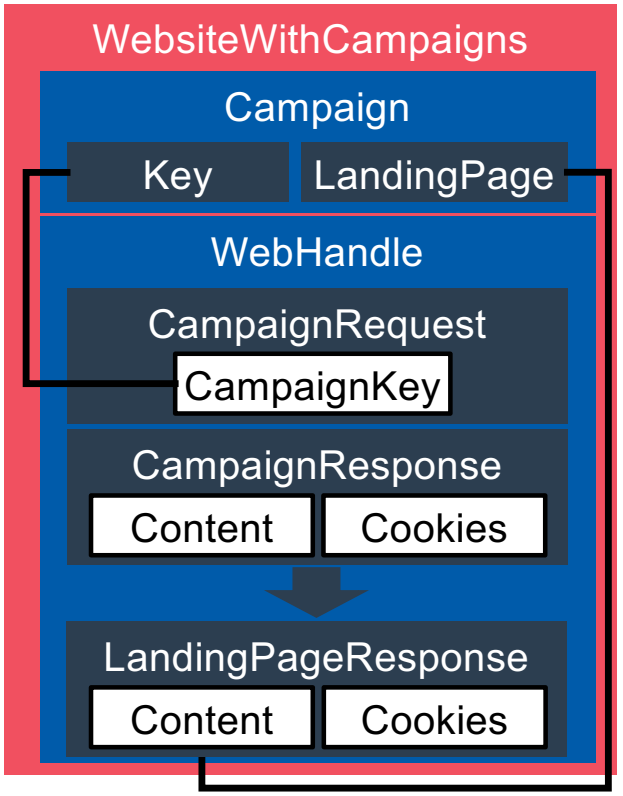
- > *Campaigns* are used to attract users and to direct their attention to certain parts of the offering.
- > If goal is to direct customers to the website:
  - > *Landing pages* as touchpoints for campaigns on the website.
  - > When arriving on a landing page, a user is assigned to a campaign.
  - > Assignment used to track the further journey accordingly, to present special offers as part of the campaign, etc.
- > For this example, we use Cookies to store the information about the campaign a user participates in
  (as opposed to the URL segment used for the language preference in the previous example).



// Basic M³L concepts for website campaigns

> Concept:

```
WebsiteWithCampaigns {
  LandingPage is a WebPage
  Campaign { Key LandingPage }
  WebHandle {
    CampaignRequest is a Request {
      CampaignKey is the Value from Cookies { campaign is the Name }
    } |= CampaignResponse {
      WebPage { URL from Request is the URL }
      from Campaign { CampaignKey is the Key }
      is the Content
      Cookies from CampaignRequest is the Cookies }
    LandingPageResponse is a CampaignResponse {
      Content is a LandingPage
      CampaignCookie is a Cookie {
        campaign is the Name
        Key from Campaign { WebPage is the LandingPage } is the Value
      } is a Cookies }
  }
}
```



// A campaign website example

> Example of a website with campaigns:

```
ACME.com-Site is a WebsiteWithCampaigns {
  Product2Page is a WebPage {
    Product2 is the Title
    Product2Description is a Content }
  BuyAProduct1Campaign is a Campaign {
    BuyAProduct1Campaign is the Key
    BuyAProduct1Page is the LandingPage {
      "Buy More Product1" is the Title }
    Product2Page {
      Product1Teaser is an Article {
        "Buy a Product1" is the Title
        "Go to Product1 page" is the Text
        Product1PreviewImage is the Image
      } is a Content } } }
```

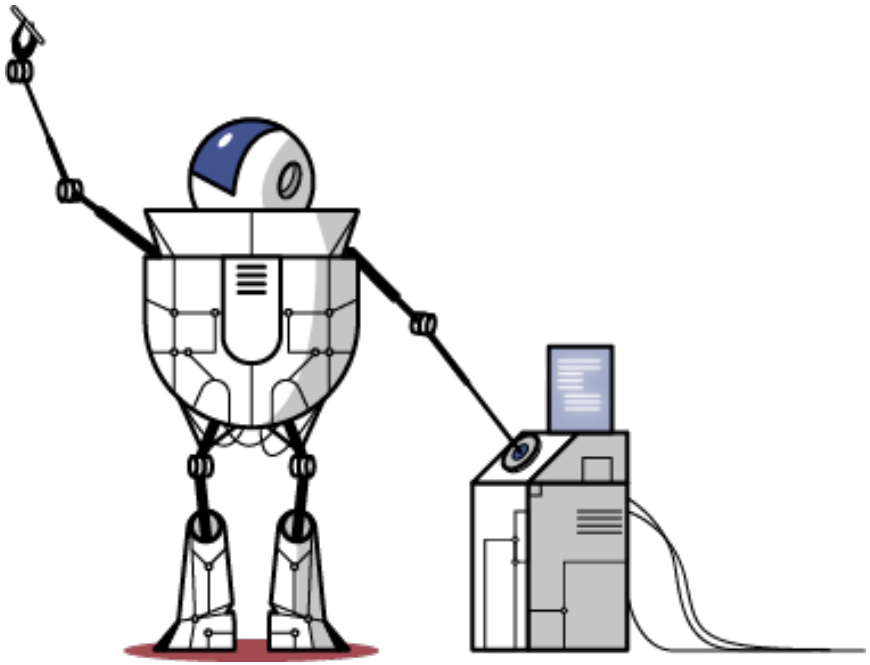
// Summary

> Many components of software system relate to an overarching model of the domain entities managed.

> Therefore, content models, like most data schemas and information models, are of central importance for a software system.

> The delivery of content in an interactive, request-based manner relates to to content in various ways through requests, request handling, and the resulting responses.

> The examples of multilingual and campaign-specific content demonstrated that content transmission can be integrated with content models for coherent interactions.



TALLENCE

THANKS.

Hamburg

Tallence AG
Neue Gröningerstraße 13
20457 Hamburg

T +49 40 36 09 35 100
F +49 40 36 09 35 200
info@tallence.com

Frankfurt/Main

Tallence AG
Walther-von-Cronberg-Platz 1
60594 Frankfurt am Main

T +49 69 401 50 76 60
F +49 40 36 09 35 200
frankfurt@tallence.com

Karlsruhe

Tallence AG
Erbprinzenstraße 23
76133 Karlsruhe

T +49 721 38 13 49 10
F +49 40 36 09 35 200
karlsruhe@tallence.com

Görlitz

Tallence AG
Neißstraße 27
02826 Görlitz

T +49 40 36 09 35 100
F +49 40 36 09 35 200
goerlitz@tallence.com

Marburg

Tallence AG
Am Grün 40
35037 Marburg

T +49 40 36 09 35 100
F +49 40 36 09 35 200
info@tallence.com