On Integrated Models for Coherent Content Management and Document Dissemination

Hans-Werner Sehring
Tallence AG
Hamburg, Germany
e-mail: hans-werner.sehring@tallence.com

***Abstract*—Content plays a central role in many information systems and commerce applications. Throughout such systems, content is dealt with in various places: storage, editing and management, quality assurance, transmission, relationship with various data, etc. Typically, software provides according functionalities in a generic way for all content applications, but they each heavily rely on a content model and need to be mapped to content individually. An alternative approach is to instead define all content utilizations in conjunction with the content model so that they are coherent throughout the whole system by design. In this paper, we study such coherent models by considering request-based document transmission that underlies, e.g., the World Wide Web. For these cases, we demonstrate that such a model-based approach is feasible, and that a configuration of generic functionality can be derived from the integrated models.**

***Keywords*—content management; content publishing; content distribution; web publishing; communication protocol.**

I. Introduction

Websites and mobile apps are an important user interface to information sources, and they constitute a means for companies to get in touch with their customers. These applications are based on content that is presented to users in a suitable form.

Contemporary implementations of *Content Management Systems* (*CMSs*), online shops, campaign management solutions, and other content-based systems deal with content in various places: databases, application code, user interfaces, remote calls, URL formatting, HTTP request handling with content lookup and caching, tracking, targeting, campaign management, and many more. Figure 1 gives a rough overview.

Each of the indicated functionalities is related to the underlying content model, and all share a common notion of both this model and all content constellations. Or, viewed the other way round, a content model defines multiple interfaces that are consumed by different audiences, for example in a CMS:

- editors that are guided by the editing tool when entering content into forms, and that are supported during quality assurance of webpages,
- application programmers that customize the CMS (services, editor, search engine),
- application programmers that develop client-side apps (JavaScript apps, mobile apps), and
- template programmers that implement the rendering of content into documents.

In fact, each of the roles uses more than one interface, and all need to agree on the conceptual content model in order to use and serve the interfaces correctly. In particular, content needs to be encoded as data, and data need to be interpreted as content.

Therefore, for coherence in content-based applications, there is a need for central models that are consistently implemented, or schemas and code for such systems are generated from such models.

In this paper, we study the process behind content distribution, typically consisting of requests for documents, the generation of the requested documents from content, and their transmission. To this end, we start in Section II by introducing our basic notions of content, documents, and document delivery. We use the *Minimalistic Meta Modeling Language* (*M³L*) to discuss the integration of transmission protocols into content models. As a foundation, Section III briefly introduces the concepts of the M³L and its application for content management tasks. As part of our experiments, Sections IV and V present models for multilingual websites and for website campaigns, respectively. The paper closes in Section VI with a conclusion and an outlook on future work.

II. Content Management and Delivery

Before discussing content presentation and shipping, we introduce basic content management terms.

A. Content

Content, in the digital domain, means the content of digital publications: (meaningful) texts, images, videos, etc. Typically, the overall set of content is composed of single *content items* that may be structured in themselves. Often, the term *asset* is used for content that cannot be divided further: a smallest text fragment, an image, and similar.

Content management strives to store content in a neutral form that allows a versatile use: in a reusable way, so that content can be combined in more than one way; media-agnostic, so that it can be used for publications on different channels; internationalized, so that it can be applied in multi-language environments; etc.

Typically, other data and information is managed on top of content, e.g., for a website:

- description data and metadata used to manage content
- navigation structures by which content is organized
- redirect rules to direct web accesses in certain cases
- header information of webpages to direct search engines, external caching layers, and other external systems

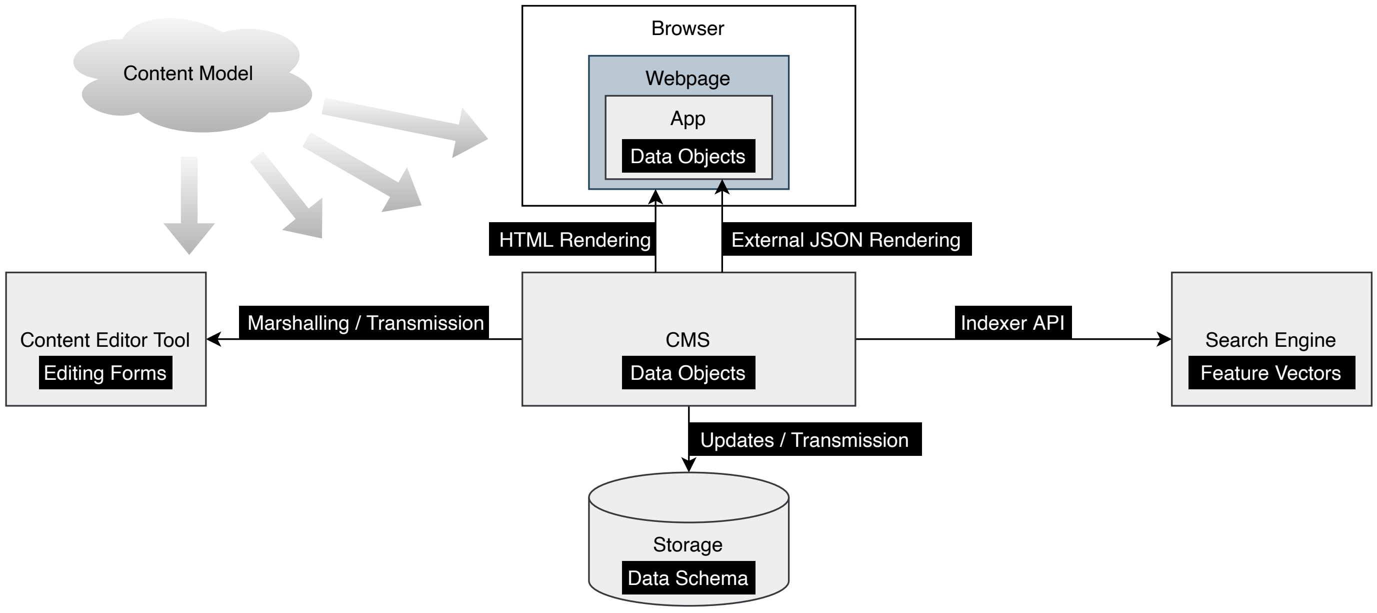

Figure 1. Content model references in a typical content management system.

B. Documents

Documents are presentations of content used for publication. They are created by applying a layout. The process of document creation is called *rendering*. In many cases, it is based on *templates* that implement the layout and declare at which positions in a layout content is filled in.

Documents may be rendered in a uniform way, or they are tailored to the needs of individual users or user groups. The individual preparation of documents is called *personalization* or *targeting*. It is based on content items that are related to certain topics and rules for the selection of items for a document. These are evaluated for a classification of users called user segments [1].

C. Document Transmission

Dependent of the communication channel used, the documents are shipped to users. In the case of the *World Wide Web* (*WWW*), HTTP(S) requests are received, and documents in the form of an HTML resource are sent in response. With so-called headless CMSs becoming popular, content may also be transmitted in the form of a data file, e.g., in JSON format, and be interpreted on client-side.

For personalization, requests need to provide some context information on the user and the environment. This information is used in the rendering process. Examples are:

- language / country / locale for multilingual content
- device or network information for adaptive rendering
- users' interests, active campaigns, or similar for targeting

D. Document Requests and Their Relation to Content

In order to identify a document and to provide context, requests for documents consist of content in themselves. For example, a URL contains a path, query attributes, header attributes, and a request body. Plus, the request's content has a close relationship to the actual content that is queried, very much as in information retrieval [2], or like a correspondence in schema mapping [3]. Typical examples are:

- IDs of content (items)
- content structure paths or navigation contexts of content
- (context) parameters to content (variants)

Because of these associations, there is a close connection to the content model. Consequently, requests and response formats have to be defined in accordance to a content model for coherence. Content-based software has to be developed with respect to the content model in all respects, including document distribution.

Software components can, to a large degree, be generated from such a content model. Typically, functionality is fixed in many components, and code only varies in the way that input and output is mapped to content. This mapping is based on the content model. Previous work proved the feasibility of a generative approach for large-scale applications [4].

III. Content Management by the Example of the Minimalistic Meta Modeling Language (M^{3}L)

For the discussion in this paper, we use the M³L since it proved to be a suitable language for the modeling of various aspects of content management. To this end, we briefly introduce the M³L, and we present some exemplary base models for content management and for publication on the WWW. These base models will be used for the case studies of multilingual content and of campaigns.

There are many more applications for multi-everything content management: multi-site (different domains operated by a central CMS), multi-brand (different presentations of shared content), personalization [1], and others.

```
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 }
```

Figure 2. M3L statements.

A. A Short Introduction into the M³L

In this section, we briefly introduce the M³L by highlighting those features that are central to the underlying experiments.

The basic M³L statements are:

- **A**: the declaration of or reference to a *concept* named *A*
- **A** is a **B**: 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; whenever *A* is referenced, actually *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 that defines how a string is produced from a concept, resp. how a concept is recognized from a string; when the representation of *A* is requested, it is produced by a concatenation of the strings produced out of *E*, *F*, and *G*; when no syntactic rule is defined, a concept is represented by its name; vice versa, an input that constitutes the name of a concept without a syntactic rule leads to that concept being recognized

If a concept that is referenced by one of the statements exists or if an equivalent concepts exists, then this one is bound. Otherwise, the concept is created as defined by the statement.

Existing concepts can be redefined. For example, with the definitions above, a statement

```
A is an H { C is the I }
```

redefines *A* to have another generalization *H* and *C* (in the context of *A*) to have *I* as its only generalization.

Every context constitutes a *scope*. A redefinition of a concept in a context is only applied in that context. When a redefinition of a concept takes place in another context as the original definition, we call that redefinition a *derivation*.

The concepts that are defined by such statements are *evaluated* when used. Evaluation means looking up or creating concepts and applying semantic rules.

Before a concept is referenced and before a statement is evaluated, all concepts are *narrowed down*. The narrowing of a concept is computed as follows:

1) The effective definition of a concept in some context is the set of all definitions in that context and all of its base contexts (transitive).
2) 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*.

Figure 2 shows some examples of M³L statements. Given those sample definitions, the result of an additional statement **Person** { **Peter** is the **Name 42** is the **Age**} is narrowed down to *PersonPeter* since *PersonPeter* is specialization of *Person* and its whole content matches. The statement **Person** { **Mary** is the **Name 42** is the **Age** } is not narrowed down further. It does not match *PersonPeter* since *Name* has a different specialization, and it does not match *PersonMary* since it has no matching content concept called *Age* or *42*.

B. Basic Content Management

The M³L is universal and has many applications. Amongst other modeling tasks, it has proven useful to describe content as characterized in section II-A. This applies both to content models as well as content items since the M³L does not distinguish model layers such as type and instance.

For example, with a content model like:

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

according 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 }
```

C. Basic Document Rendering

For textual formats, like for example HTML, documents can be rendered from content through syntactic rules of content as declared in the previous subsection. On the level of the content model, syntactic rules describe document templates, on the content item level they render single document instances.

A sample template for *Article* from the previous subsection is:

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

The syntactic rule defines an HTML structure into which the concepts from the content are integrated. These may themselves evaluate to content strings of embedded HTML structures.

Please note that, e.g., *</div>* is a valid concept name, as is *class=\"article\">*. Since new concepts are declared the first time they are referenced, and because they syntactically evaluate to their name by default, they can be used like string literals. The concept name \" is an escape sequence for the quote character (not a quote sign for identifiers).

In this example, *ImageResource* may be a concept of an image resource that has a *URL*.

```
01 URL { Protocol Host Port Path }
02 |- Protocol :// Host : Port Path .
03 WebPage { Title Content URL }
04 |- <html>
05    <head> <title> Title </title> </head>
06    <body> Content </body> </html> .
07 Cookie {
08  Name is a String
09  Value is a String }
10 |- Cookie: Name = Value
11 Request {
12  URL ProtocolVersion Method
13  Parameters HeaderAttributes
14  Cookies is a Cookie Body }
15 |- Method " " Path from URL " "
16    Protocol from URL / ProtocolVersion
17    \n HeaderAttributes \n Cookies \n \n
18    Body .
19 ResourceResponse is a Response {
20  Protocol ProtocolVersion Cookies
21  Content }
22 |- Protocol / ProtocolVersion " " 200
23    " " OK \n Cookies \n \n Content \n .
24 WebHandle {
25  2 is the ProtocolVersion
26  Request {
27   ...
28  } |= Response {
29   WebPage {
30    URL from Request is the URL
31   } is the Content
32   Cookies from Request is the Cookies }
33 }
```

Figure 3. Basis HTTP concepts in M3L.

D. Basic Document Delivery

To study the interplay of content and document transmission protocols, we also sketch possible models for the latter. In particular, we use a simple subset of HTTP because it is the protocol of the WWW, and it is a text-based protocol with a simple grammar and, therefore, easy to model with the M³L.

Figure 3 shows a set of concepts representing HTTP.

Documents become (web) resources by making them available under a URL. A concept *URL* is modeled in a simplified form after RFC 1738 in lines 1–2. Further concepts may be introduced to configure a *URL*, e.g., `HTTP is a Protocol`.

With URLs, webpages may in a very simple form be modeled by the concept *WebPage*. In this example, they just have a page title, the content displayed on a page, and the URL under which to reach the page. In real-world applications, the URL is generated from, e.g., the website's navigation structure instead of being maintained explicitly. This simple form is sufficient for the discussion of this paper, though.

```
01 MultiLingualWebsite is a Website {
02  Language { Code }
03  URL { Code } |- Protocol :// Host :
04                  Port / Code Path .
05  WebHandle {
06   Request |= Response {
07    WebPage {
08     URL from Request is the URL }
09    from Language {
10     Code from URL from Request
11      is the Code }
12    is the Content } }
13 }
```

Figure 4. Basic M³L concepts for multilingual websites.

Requests to a URL are answered by a web server with a response that contains the addressed resource. Such a web server can in essence be modeled by the concept *WebHandle*. It is a base concept to derive web servers from.

For the construct to work, we assume a M³L runtime that accepts input coming from network connections and recognizes it in the context of the *WebHandle* at hand. When the input is recognized as an HTTP request, then a *Request* concept will be constructed from it. Once the *Request* was created, its semantic rule will fire, thus finding or creating a *Response*. In case a web page has been found, this will be a *ResourceResponse*. (We omit all error handling and error responses.)

The response is created with that webpage as content that has the requested URL assigned. This is achieved by the matching in lines 29–31 of Figure 3. Cookies are echoed from the request, so that they are sent back to clients (line 32).

Of course, HTTP contains many more elements. Here we just picked those ones that we need in the course of this paper.

The syntactic rule of *Request* is used to recognize requests, that one of *ResourceResponse* to create the response output. Let the concept $\backslash n$ be defined to syntactically produce a line break in this example.

IV. Multilingual Websites

As the first case for the study of the integration of a content model and a request protocol, we consider a multilingual website. For the purpose of this paper, we only consider localized content; a very basic form of localization [5]. Other kinds of multisite platforms like websites in different countries, for different brands, etc. may be modeled in a similar fashion.

Assume a website with multilingual content and webpages that are requested by URLs that have a language code as their first path segment. This is a typical example of content and access protocol being related: the first path segment of a URL uses a language code that is also used to select a content variant, and the remaining path of a URL identifies the content.

Figure 4 shows base definitions for such kinds of websites. For each multilingual site, a language context is defined based on *Language* in line 2. The language context is identified by a

```
ACME.com-Site is a MultiLingualWebsite {
 WebPage4711 is a WebPage {
  URL4711 is the URL {
   ... /bignews 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 } }
 }
}
```

Figure 5. A sample multilingual website.

language code, and translated webpages will be defined within a language context.

URLs are redefined for multilingual websites so that they include the language code.

The *WebHandle* is changed for multilingual websites so that it evaluates the language setting in the URL path. It establishes the relation between URLs and content. Requests use the redefined *URL*s that recognize the language code. The *Content* of a *Response* is assigned a webpage that is retrieved from the corresponding language context: the *Content* is assigned the webpage with the matching *URL* (line 8 in Figure 4) resolved relative to the *Language* context with the given *Code* (lines 9-11). The *URL* is matched without considering the *Code* when it is not contained in the *URL*s assigned to webpages.

An example of an application of this base model for multilingual websites is shown in Figure 5. It shows a basic sketch of a website that is available in two languages. One page, *WebPage4711* is defined with its *URL* and an *Article* as its *Content* that contains one (language-independent) image.

In the two language contexts, translated derivations of the webpage are defined. These have translated titles and article texts.

Since the whole website is derived from *MultiLingualWebsite*, URLs with paths */en/bignews* and */fr/bignews* will be recognized and resolved to the two translated webpages.

V. Campaign Management

As part of modern digital marketing, campaigns are used to attract users and to direct their attention to certain parts of the offering. Campaigns originate in many places, also outside a website or even any digital channel. One of their goals may be directing customers to the website, though. In that case, a

```
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 {
   WebPage is a LandingPage
   CampaignCookie is a Cookie {
    campaign is the Name
    Key from Campaign {
     WebPage is the LandingPage }
    is the Value }
   is a Cookies }
 }
}
```

Figure 6. Basic M^{3}L concepts for website campaigns.

website typically offers touchpoints for campaigns in the form of *landing pages*. Such pages welcome the user to continue the *customer journey* one the website.

When a user enters a website through the landing page of a campaign, then the customer journey continues in the context of that campaign. Assigning the user to a campaign can possibly be used to track the further journey accordingly, to present special offers as part of the campaign, etc.

Tracking campaigns can be based on a key, similar to site languages that have a code in the example in the previous section. In a sample implementation considered in this section, a visit to a landing page leads to a *cookie* with the campaign key being set.

A base model for a site with this behavior is shown in Figure 6. For such a site, campaigns are contexts derived from *Campaign* that have a unique *Key*. In this context, campaign-specific webpages may be defined.

The page variants in the context of the campaign may, e.g., define the page with different content that relates to the campaign. If no derived page is defined in the context of the campaign, the original page from the outer context of the website will be used.

On top of that, each campaign defines a webpage as a *LandingPage*. When a user visits such a landing page, a cookie

```
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 }
 }
}
```

Figure 7. A campaign website example.

with the corresponding campaign key is added to the request. This is performed by a specific web handler.

The web handler for campaign tracking has to deal with two cases:

1) When the user accesses a *LandingPage*, then a cookie that identifies the campaign is set.
2) When a user accesses any other page, then existing campaign cookies are maintained.

This behavior is achieved by narrowing the response that the web handler generates.

For campaigns, we add a derived kind of request that extracts a campaign key (if one exists) from a cookie with the name “campaign”. The webpage delivered in the response *CampaignResponse* is the webpage with the requested URL. The lookup of webpages originates in the context of the campaign, though, in order to allow campaign-specific webpages. Cookies, in this case, are just echoed from the request.

Additionally, there is a specialized response for the case that the resulting webpage is a landing page. Through the narrowing of concepts, this derived response is used for landing pages. Other than the *CampaignResponse*, the *LandingPageResponse* sets the cookie *CampaignCookie* for the campaign to which the landing page belongs. This is done by looking up the campaigns that name the webpage as their *LandingPage*. The cookie’s name is “campaign”, the one which *CampaignRequest*s look for.

Figure 7 shows an example of a campaign tracking website based on *WebsiteWithCampaigns* from Figure 6.

An access to the landing page *BuyAProduct1Page* will lead to the campaign cookie being set. When a user accesses the page *Product2Page* with that cookie being set, then this page will be presented with an additional *Product1Teaser*.

VI. Summary and Outlook

The paper concludes with a summary and an outlook on future work.

A. Summary and Conclusion

Content models, like most data schemas and information models, are of central importance for a software system since many components of the overall system relate to them. In particular, 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.

Using the examples of multilingual and campaign-specific content, we demonstrate that the details of interactions during content transmission can basically be included in content models. In the examples given, this would be the utilization of the users’ language preferences and the consideration of customer journeys that relate to a campaign.

We furthermore deduce from the studies presented in this paper, that a content modeling language that has a notion of context eases the implementation of systems from content models since it allows expressing details on content variants and their relationships.

B. Outlook

The examples in this paper have been checked by the M³L environment that evaluates all statements given. For real-world applications, it will most probably be necessary to generate dedicated software components from the content model. This has been studied before for database applications [6] and for *Concept-Oriented Content Management* (*COCoMa*) systems [7]. Future work will investigate how such a software generation approach can be transferred to M³L statements.

Acknowledgment

Numerous clients, partners, and colleagues allowed the author to participate in exciting projects and to gain the insights that led to this paper. The author thanks his employer, Tallence, for supporting his research activities.

References

[1] H.-W. Sehring, “An integrated model for content management, topic-oriented user segmentation, and behavioral targeting,” International Journal on Advances in Software, vol. 12, 2019, pp. 372–383.

[2] P. Ingwersen and K. Järvelin, The Turn: integration of information seeking and retrieval in context, vol. 18 of The Information Retrieval Series. Springer Science & Business Media, 2006.

[3] F. Legler and F. Naumann, “A classification of schema mappings and analysis of mapping tools,” in Datenbanksysteme in Business, Technologie und Web (BTW 2007) – 12. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme“ (DBIS), A. Kemper, H. Schöning, T. Rose, M. Jarke, T. Seidl, C. Quix, and C. Brochhaus, Eds. Bonn: Gesellschaft für Informatik e. V., 2007, pp. 449–463.

[4] J. W. Schmidt, H.-W. Sehring, M. Skusa, and A. Wienberg, “Subject-oriented work: lessons learned from an interdisciplinary content management project,” Proc. Fifth East-European Conference on Advances in Databases and Information Systems, ADBIS 2001, pp. 3–26, September 2001.

[5] H.-W. Sehring, “Content structures, organization, and processes for localized content management,” International Journal on Advances in Software, vol. 10, 2017, pp. 211–220.

[6] S. Lazetic, D. Savic, S. Vlajić, and S. Lazarević, “A generator of MVC-based web applications,” World of Computer Science and Information Technology Journal, vol. 2, 2012, pp. 147–156.

[7] H.-W. Sehring, S. Bossung, and J. W. Schmidt, “Content is capricious: a case for dynamic system generation,” Proc. 10th East European Conference on Advances in Databases and Information Systems, ADBIS 2006, pp. 430–445, September 2006.