Dockets: A Model for Adding Value to Content

Joachim W. Schmidt and Hans-Werner Sehring

Software Systems Institute (AB 4.02)
Technical University Hamburg-Harburg
http://www.sts.tu-harburg.de

Abstract. Dockets[1] are traditional paper means for supporting specific classes of workflows which concentrate on content and its value for individuals and organizations. A docket prescribes and finally documents some content reviewing process which is part of a larger value chain.
Since contents represent essential investments they are carefully protected by principles like autonomy and ownership. However, value chains over contents materialize only in open and cooperative scenarios. Therefore, dockets aim at mediating carefully between open content availability for value-adding processes and control for investment protection.
In this paper we present *dockets* as an innovative *conceptual model* for investment-protecting value chains over content. The model is introduced essentially by graphical means and is substantiated by our first experience with docket-driven application development. We demonstrate how our basic docket model scales to support general content management.

1 Introduction

Information is an increasingly valuable resource for both information providers and consumers. In the rapidly extending market for networked information services, information resources represent the essential value and investment of many enterprises. Owners of information resources, individuals as well as organizations, become more and more conscious to whom and under which conditions they provide access to their content. Resource autonomy is a highly ranked quality.

On the other hand, most information resources become and remain valuable only if they are subject to regular use and to value-adding processes. However, since such efforts may severely interfere with resource autonomy and ownership, they have to be realized by carefully monitored and agreed-upon processes. Furthermore, adding value to content nearly always requires some deeper insight into the larger value chains in which the resource and its content is involved.

[1] Docket
- a piece of paper accompanying or referring to a package or other delivery, stating contents, delivery instructions etc.;
- a summary of contents, as in a document;
- a list of things to be done;
- sometimes serving as a receipt.

[Collins Dictionary of the English Language, Collins London & Glasgow, 1981]

As a consequence, there is an increasing demand for support of a specific class of workflows which implement investment-protecting value chains over content. Such content-oriented value chains have to mediate carefully between possibly conflicting interests of the contributing domains:

- reliable content protection as demanded by resource content providers,
- liberal resource access as required by content reviewing experts, and
- goal-oriented exploitation of reviewing results by value chains over content.

In this paper we concentrate on the conceptual modeling [2] of those workflows intended to add value to content. For this purpose we analyze, abstract and apply *dockets*, which is a proven concept from traditional content management. Offices and organizations use paper dockets if they want to initiate and finally document some routine, *re-view*-like operation on a piece of content which is attached to or referred by the docket. Furthermore, each docket addresses some review performer by whom it is returned to sender upon review completion.

In section 2 we motivate the concept of dockets and sketch a model by giving some of the major abstractions provided by dockets.

Section 3 discusses docket dynamics in terms of performer binding, property handling and docket tracing. In section 4 we apply our basic docket model to an extended range of applications.

The docket life cycle is discussed in section 5. First experience with docket-driven application development is reported in section 6 together with the emerging architecture of a generic framework for docket management. In section 7 we refer to related work before we make some concluding remarks in section 8.

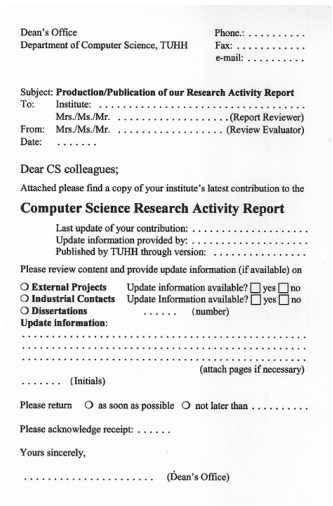
2 Dockets as a Conceptual Model

Traditionally, dockets are used for those (parts of) value chains in content management which follow well understood and somehow repetitive content-oriented interaction patterns. Typical docket examples are version update and management of reports, classification and annotation tasks on library material, or management and reviewing of conference submissions.

2.1 Dockets: A Proven Concept

In many cases content providers give access to their material only if they understand and agree why, how and by whom the content is used. On the other hand, reviewers working on contents and possibly contributing to their value expect a clear insight into the larger value chain within which their result will be evaluated and exploited. It is essential for content management to clarify the assumptions which performers participating in a docket scenario make about their roles and mutual relationships. Docket design makes such assumptions and roles explicit, and docket enactment supports their achievement.

Fig. 1(a) shows an example of a paper docket *template* for the periodic reviewing of our department's Research Activity Report. Docket templates have to meet *by design* the interests of all three domains and their performers:



Dean's Office
Department of Computer Science, TUHH

Phone.: .....
Fax: .....
e-mail: .....

Subject: **Production/Publication of our Research Activity Report**
To: Institute: .....
Mrs./Ms./Mr. ..... (Report Reviewer)
From: Mrs./Ms./Mr. ..... (Review Evaluator)
Date: .....

Dear CS colleagues;

Attached please find a copy of your institute's latest contribution to the

Computer Science Research Activity Report

Last update of your contribution: .....
Update information provided by: .....
Published by TUHH through version: .....

Please review content and provide update information (if available) on

❍ **External Projects** Update information available? ☐ yes ☐ no
❍ **Industrial Contacts** Update Information available? ☐ yes ☐ no
❍ **Dissertations** ..... (number)
Update information:
.....
.....
.....
(attach pages if necessary)

..... (Initials)

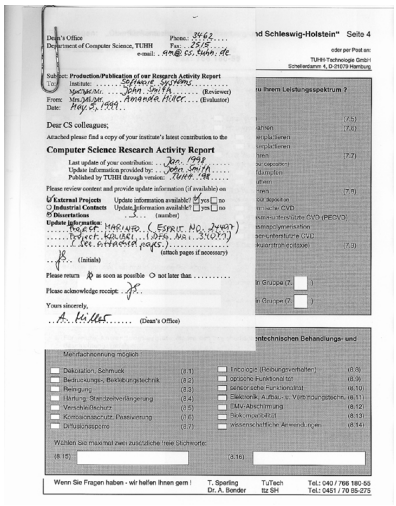
Please return ❍ as soon as possible ❍ not later than .....

Please acknowledge receipt: .....

Yours sincerely,

..... (Dean's Office)

(a) Paper Docket Template for Content Review



Dean's Office
Department of Computer Science, TUHH
Phone.: 3462
Fax: 2515
e-mail: am@cs.tuhh.de

Subject: **Production/Publication of our Research Activity Report**
To: Institute: Software Systems
Mr./Ms./Mr. John Smith (Reviewer)
From: Mrs./Ms./Mr. Amanda Miller (Evaluator)
Date: May 5, 1999

Dear CS colleagues;

Attached please find a copy of your institute's latest contribution to the

Computer Science Research Activity Report

Last update of your contribution: Jan. 1998
Update information provided by: John Smith
Published by TUHH through version: TUHH-98

Please review content and provide update information (if available) on

☑ **External Projects** Update information available? ☑ yes ☐ no
❍ **Industrial Contacts** Update Information available? ☐ yes ☐ no
☑ **Dissertations** 4 (number)
Update information:
Project MARVEL (ESPRIT No. 2443)
Project KOLUMBUS (DFG No. 3/0077)
1 new dissertation pages).

JS (Initials)

(attach pages if necessary)

Please return ☑ as soon as possible ❍ not later than .....

Please acknowledge receipt: JS

Yours sincerely,

A. Miller (Dean's Office)

(b) Paper Docket Instance with Content Binding

Fig. 1. A Paper Docket

- *content providers* have to understand the utilization of their content so that they can decide whether and how they want to contribute;
- *content reviewers* need a clear image of their task so that they can judge the purpose of the entire action and the role of their contribution; and finally,
- *review evaluators* at the dean's office want to represent their interest by prescribing docket property types which are to be filled in by the reviewers.

In our example the department sees added value in publishing an up-to-date and extensive Research Report. Therefore, the dean's office periodically initiates a reviewing process by issuing docket instances and binding them to both existing content and to the responsible reviewers. Resource autonomy is maintained by attaching only a copy of last year's content.

Fig. 1(b) refers to an example of a paper docket *instance* created by enactment of a review docket based on the template of fig. 1(a). This docket instance initiates the reviewing of the currently published Research Report of the Software System Institute and finally reports the reviewing results back to the dean's office. For this purpose, the docket instance establishes concrete bindings to the

- review content (*Attached please find a copy of ...*), and to the content provider (*Published by TUHH through version ...*);
- review results required by the dean's office (*...on External Projects ...*), and to the review evaluator (*From: Mrs. Amanda Miller; Phone: 3462*),
- review procedure (*provide update information ...*), and to the content reviewer (*To: Mr. John Smith*).

2.2 Docket Abstractions: Templates and Their Domains

Conceptually we define dockets as a compound of three abstractions where one represents the document providing domain, a second one the domain of content reviewing, and the third one the domain of review evaluators and their value chains. Abstracting over traditional paper docket examples leads to a conceptual docket model with a common docket template structure as presented in fig. 2.

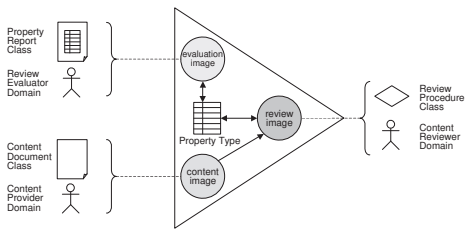


Fig. 2. Docket Templates and Domains

While the template level refers to domains, at the instance level concrete bindings to instances out of those domains are established and maintained: a content providing document is attached, a reviewing procedure instance is bound, and properties are specialized for content reviewing and review evaluation.

On the one side, dockets *correlate* aspects of three different *domains*, on the other side, they *separate* the issues of three distinct *concerns* – docket design, presentation, implementation. It is essential to docket modeling to understand how both groups of tasks, domain correlation and concern separation, are distributed over the two *levels* of docket templates and docket instances. As a consequence, we concentrate on issues of docket design, and we discuss these issues on the two levels of docket templates and docket instances.

In this paper we leave implementation issues to a minimum. We want to emphasize, however, that one of the prime objectives of docket design is to enable a coherent and meaningful presentation ("best docket images") of the three domain abstractions involved in each docket.

Content images on the docket template level inform potential content providers about the requirements expected by content instances at docket binding time. This might include type information as well as requirements on content-oriented quality, access rights, format, layout, space, time etc. The weaker requirements are based on copy semantics, reference semantics usually imposes more restrictions on content autonomy.

The example in fig. 1 shows that a substantial part of the docket is used to provide an image of what the review is all about. Though reviewers are autonomous, the **review image** may impose specific constraints, e.g., by prescribing the order in which the content is to be read. Additionally, qualifications on the "career record" may be imposed on reviewers. Technical capabilities required by a performing reviewer can also be specified.

Evaluators are supposed to exploit the review results in the light of the encompassing value chain. They therefore need to communicate its relevant aspects by the **evaluation image**. This might include making additional aspects explicit: the evaluator's identity and intent, the overall value chain, the identity of the docket initiator, etc. Evaluators express their demands on docket results essentially in terms of properties and their types. This is discussed in section 3.2.

3 Docket Enactment: Instances and Their Bindings

In the previous section we discussed docket templates and the consideration of the three domains involved. Fig. 3 depicts a docket instance with established bindings. Like type definitions in formal programming models, template information essentially serves the purpose to make domain assumptions explicit at docket design time and to assure that at docket enactment time only instances are bound which meet those assumptions.

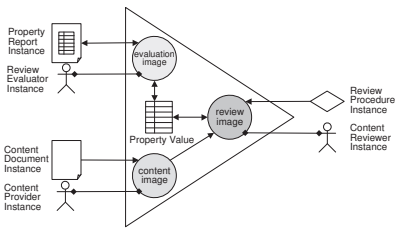


Fig. 3. Docket Instance and its Bindings

3.1 Docket Enactment

When a (step of a) value chain is to be carried out under docket control, an appropriate docket template has to be found or designed. From templates, instances are created, and a generic process scheme is invoked. As with workflow modeling we call this process scheme docket enactment. A generic docket enactment scheme is sketched by the UML activity diagram in fig. 4.

Binding docket performers starts with selecting individual entities and performers from each of the three domains. The initiative for docket enactment is often taken by the person performing as an evaluator. There may, however, be cases where the content provider, the reviewer, or a third party initiates docket enactment. The selected domain entities are checked against the three corresponding images defined by the docket template. One performer might base his decision about taking part on the choice of the others (not shown in fig. 4).

By docket binding, the content is provided, the properties are initialized, and the review procedure is started. In general, each domain contributes a static part provided by the image, and a dynamic part provided via performer binding.

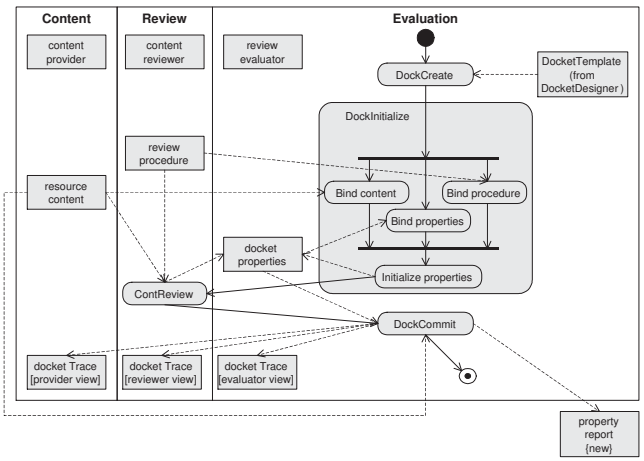


Fig. 4. Generic Docket Enactment Scheme

Interpreting docket modeling as a contracting activity, docket enactment can be seen as the signing of a contract. Docket images should enable all three performers to understand the overall purpose of the docket which, finally, all three have to agree and to accept. This may require further negotiation. Once the performers are bound to the docket instance they are bound by the "contract" [28].

3.2 Docket Properties and Traces

Properties serve as the basis for defining and achieving the added value through a docket: on the template level, properties and their evaluation image context communicate the goal of the review, on the instance level, properties take up the review results. After a docket is terminated properties are stored in a *property* or *review report*. This is all the evaluator gets out of a docket enactment.

We refer to property specification on the template level as the property type. It includes the structure of the properties, the range of possible values, the units and metrics in which to measure them, constraints on the values' accuracy, tolerance etc. In the most general case of human reviewers, a property specification might be given in natural language (compare fig. 1). In the most specific case, a type definition for some software component will be provided.

By defining and setting properties, the evaluator influences the review process. The values of the initialized properties are input for the reviewer additional to the property type. Under extended protocols, the evaluator may observe the reviewer at work. If the review is not likely to satisfy the evaluator's expectation, he may adjust the properties to guide the course of the review.

As a value container for results, the properties are implemented by some data or document model. In our docket framework (see section 6) we experiment with SGML and XML [11], exploiting the possibility of mixing dynamic and static content in a defined structure. An important advantage is the possibility to

provide parts of documents as valid documents; for XML "fragment interchange" is currently being discussed [8]. Advanced property models will require extensions for typed attributes with metrics, tolerances, taxonomy extension etc. [1].

Often the relationship between the property structure and those parts of the document which provide the content has to be preserved. If the content structure is known to the reviewer and can be encoded in the properties, the mapping can be implemented by cross-references. In our docket framework, both properties and contents are given as documents with known DTD, and we use links defined by the XML Linking Language for cross-referencing [15]. If direct references from one model to the other are not possible, annotation models might be appropriate [27]. In such cases properties are annotations that are attached to those parts of the content which they comment.

At docket commit time, any binding between the three docket contributions is released. However, since docket enactment is a value in itself to any enterprise as well as to the performers, each enactment is documented. The three performers can expect at least two benefits from such a documentation. First, it serves as a receipt of their participation in some docket enactment, be it for controlling purposes or for proving that they fulfilled the contract defined by the docket. Second, they might use it as a measure of their own professional performance.

In our docket framework each enactment produces a trace. We experiment with various trace and traceability models from software engineering [23], requirements engineering [25, 5], and workflow history management [14]. Our approach to traces is property-based, i.e., they are in essence versioned and materialized views on the properties and the review procedure. For a detailed discussion of the relationship between document, version, and process models see [12].

For added organizational value, traces are stored in a database [22] maintained by the docket framework. It allows queries on the trace extent which provide a basis for data warehousing over content-oriented value chains.

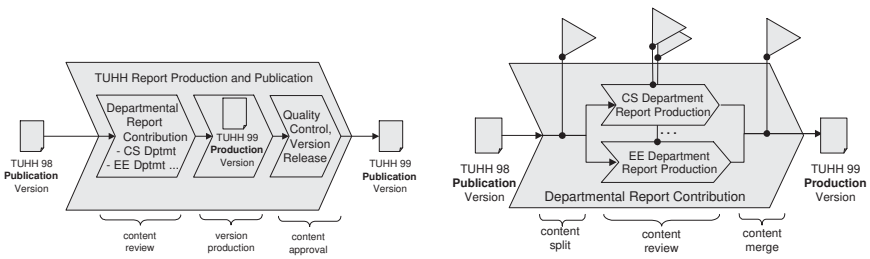
Each docket performer may get a specific variant of the trace and may use it at will – with one notable exception. To serve as a proof in the other performers' eyes, traces have to be immutable. This is the basis for acceptance in a legal context or as a measure for the reviewer's professional performance.

4 Extended Docket Scenarios

Our basic docket model presented so far is based on the three principles: resource protection by controlled access, content review by reviewer assignment, and evaluation through property-based value chains. Based on our first experience with docket-driven application development we demonstrate how the basic docket model can be used to support more general content management schemes.

Patterns of particular interest are those which span over multiple dockets. They are required, for example, in cases where results of some initial review are used to create new document versions.

The docket from the example of fig. 1 aims at producing the annual version of the CS department's part of the of the Research Report. The overall value chain



(a) A Value Chain (b) Value Chain with Dockets

Fig. 5. Value Chains in the Research Report

is modeled by fig. 5(a): it updates the 1998 document version and publishes the 1999 edition. For further discussion, we refer to the "fish" notation (or "Chevron" symbol) which is also used in SAP's reference model [6] as an informal mean to name, structure and relate value chains.

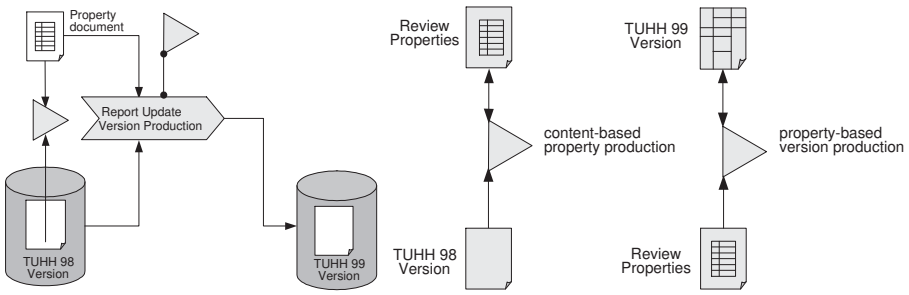
Value chains are composed recursively. Fig. 5(a), for example, shows three sub-chains. Starting point of the value chain is the published version of the Research Report. The Production Version constitutes an intermediate state in our value chain, the Publication Version is its final result.

Value chain refinement reaches the docket level as soon as added value can be implemented by routine review and evaluation of content. Fig. 5(b) shows a refinement of the first member of the value chain into parallel activities. It is assumed that the deans' offices know how to implement their value chain by routine reviewing of existing contents. In fig. 5(b) this is symbolized by "raised docket flags". This connects the example back to our initial discussion (see fig. 1).

Two other parts of the value chain in fig. 5(b) are candidates for dockets, content split and merge. Research Report structure and organizational structure are fairly isomorphic, as to be expected with universities and their departments, so they can be carried out by dockets with little or no human intervention.

So far, our example concentrates on dockets for report reviewing which produces properties and leaves the initial content unchanged. The central step in the value chain which produces a report update version (see fig. 6(a)) has not yet been addressed on the docket level. When designing update dockets for version production one observes an interesting inversion of roles between property and content. As indicated by fig.6(b), the document by which the *content* for an update docket is provided, is produced and probably owned as a *property* report by some reviewing step further up in the value chain.

The deeper reason for making version update a two-step, three-state value chain lies in the fact that in professional content management one has to distinguish carefully between the three cases of ownership, on one hand – initial content, update information, update version – and, on the other, the two value-adding processes and their responsibilities – content reviewing and update.



(a) Property-based Version Production

(b) Properties and Documents in Version Production

Fig. 6. Dockets for Version Update

5 Docket Lifecycle

There are clearly distinct phases which determine the lifecycle of a docket. The first and most important phase is problem analysis which is based on content-oriented use cases and value chains; this phase results in the design of a docket template. Two additional phases are the phases which deal with docket presentation and docket implementation.

5.1 Docket Design

Besides rather generic patterns which are shared by most dockets, e.g., patterns for docket enactment (compare fig. 4, section 3.1), there are many more specific decisions to be made by the docket designer: specification of review procedures, property sets and access protocols through which content is provided.

Fig. 7 specifies as part of a docket design the typical tasks to be performed by a reviewer. In this case a reviewer performs his task in one go, i.e., without any interaction with the provider or the evaluator. The reviewer begins by taking notice of the content (Read Content). In fig. 7 we distinguish three alternative review procedures: The first kind expects the reviewer to just read the content (notification). For the second kind of task (ranking) the reviewer is supposed to produce a property by selecting a value from a discrete value set as some ranking. For the third kind of review procedure (annotation) properties are expected to be of a type which leaves room for the reviewer's creativity, e.g. free text, images, speech etc. Any review ends with acknowledging that the review was performed.

5.2 Docket Presentation and Implementation

Good docket *presentation* is essential to the entire approach. Its objective is to communicate by its three images the objectives of the docket at hand. The

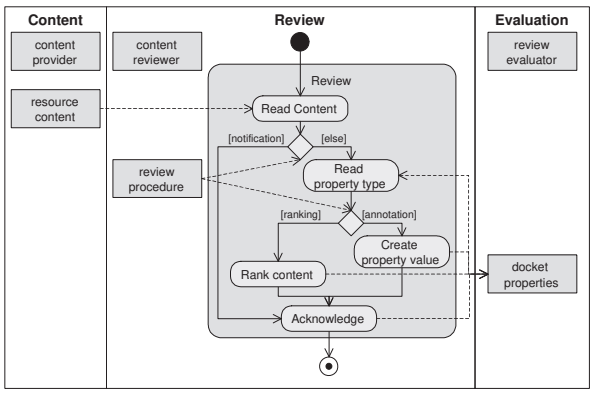


Fig. 7. Activity Diagram of Standard Review Image

partitioning into three domains which is made by the model is given up in favor of a coherent docket presentation. Our paper dockets in fig. 1 already show such coherence, the example subsequently discussed demonstrates some of the additional potential of screen-based docket presentation.

Our view on dockets is heavily influenced by experience with a digital library project with our colleagues from the Art History department at Hamburg University. The project Warburg Electronic Library (WEL) [30] deals with objects of art which are or have been used for conveying political concepts, messages and goals. In the WEL project we distinguish carefully between various signifier-signified relationships such as iconic, indexical and symbolic references [7].

The WEL provides docket-driven applications for icon classification, annotation, tracking etc. Fig. 8 shows the presentation of a WEL docket. The left-hand-side offers a selection of the ontology terms used by the WEL for image classification, the lower right-hand-side a stack of images, and the upper right-hand-side a text which specifies the classification task. The images are the content for which added-value is to be produced. Additional content is provided by a file named in the task description. This text together with the drag-and-drop layout elements constitutes the image of the review procedure. On the template level, docket properties are represented by the various fields, e.g., title ("Titel"), artist name ("Künstler"), keywords ("Schlagwort", "Stichwort"), and their types. The concept hierarchy is also part of the property type. On the instance level, properties are partially initialized (e.g., keyword "mit Tier" or artist "(Cesare Ripa)").

The university Research Report discussed in section 4 is another docket-driven application project. The paper-based procedure of updating the report has been replaced by an online system where the university owns and provides the content, the departments organize the production of new document versions, and the institutes review reports regularly and provide update information.

Besides driving docket presentation, docket design also captures the essential requirement for docket *implementation*. Since this paper concentrates on the conceptual modeling aspects of dockets only a few references to our implemen-

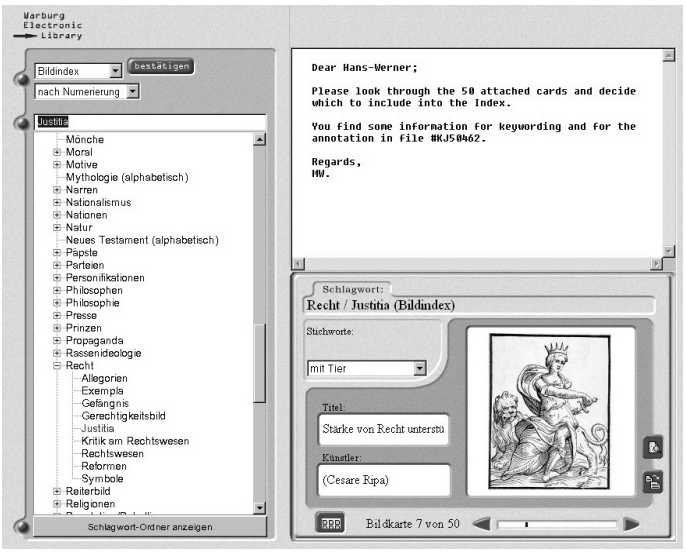


Fig. 8. Docket for Image Classification: Presentation Level

tation experience will be made. Some requirements have already been discussed above in the context of generic docket activities (compare figs. 4, 7). Our first insight into an emerging docket framework is outlined in the subsequent chapter.

6 Towards a Docket Framework

Content management patterns as discussed in sections 3 and 4 are becoming common in online content production and publishing. As a result we present the architecture of an emerging docket framework and discuss some of its components. We built upon CoreMedia, a commercial product for editorial management developed by a Hamburg-based software vendor. Work on a docket framework is considered to be a contribution towards an emerging *Docket Technology*.

For resource autonomy reasons, content providers and docket handlers (editors, producers, publishers) reside on separate servers, the former on a publication server, the latter on a production server. Demands for docket cooperation lead to the following architecture (fig. 9):

- *publication servers* present, store and protect released content;
- *production servers* have controlled access to publications and host dockets;
- *specific dockets* manage trusted content exchange between the servers;
- *generic dockets* communicate and encapsulate value-adding services.

By decoupling the two servers and their repositories, content protection and resource autonomy are established. Specific docket templates for content split and content merge are designed to control content exchange between production and publication servers and to minimize the need for human intervention.

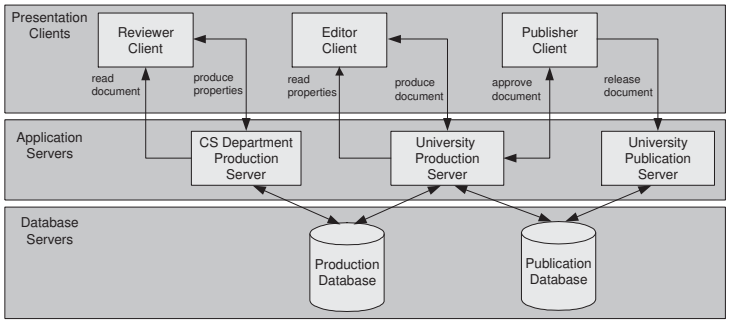


Fig. 9. System Architecture for Online Report Production and Publication

Dockets for content reviewing tasks are represented on editor clients while dockets for the more mechanical parts of content management such as document split, merge, compilation etc. reside on producer clients. Both kinds of dockets share the production server which interacts with the underlying production database. Our docket framework supports integrity management and type checking on various levels and offers a range of binding alternatives [29].

Docket presentation clients are currently implemented in two versions: the one version is a Java-based client program, the other one is a web-based, form-oriented version implemented by HTML pages and Java servlets. The specifications of the review procedure are visualized, and review operation is monitored.

The development of a generic client which can generate initial presentations based on the images of docket templates is under way. Together with the growing generic functionality provided by our docket framework, support for docket visualization will substantially speed up docket-based application development.

7 Related Work

Throughout this paper we already mentioned some links to other models. In this section we have a closer look on their relation to dockets.

7.1 Workflow Models

Both dockets and workflows are based on models and methodologies which enable enterprises to find abstract descriptions of their essential value chains, see commonalities between processes, and benefit from such insight. Both models profit from the fact that stereotypic processes can be generalized by templates.

Generic workflow models focus on the flow of control by which processes essential to the enterprise are activated and constrained. Such processes are usually distributed over organizations and require validation, optimization and control [10]. In contrast, dockets do *not* aim at representing entire value chains.

Potential docket performers have to be able to decide whether and how to participate in docket enactment. For this purpose they have to understand the

value chain behind a docket. Most workflow models have shortcomings in this respect. The Toronto school with its conceptual modeling background is one of the few exceptions: the i^* method [32], for example, pays specific attention to capturing the intent behind an enterprise's activity.

Some approaches include resource models and control instances by integrating organizational models with workflow and data models [3]. The organizational model defines the possible delegation of work. In docket modeling we assume, however, that docket performers are autonomous actors which may not even belong to the same organization.

Electronic Circulation Folders (ECFs) [24] offer a metaphor similar to dockets. They are used to clarify migration, ownership, and responsibility. ECFs address users who are comparable to reviewers, not different domains like dockets do. The main goal of ECFs is asynchronous cooperation as we address it on the value chain level. Resources are not of interest in their own.

7.2 Agents

Performers who cooperate in a docket scenario can be seen as independent agents bound by a common task to be solved. They are in a multi-agent setting observing each others behavior. An evaluator, for example, may observe a reviewer to make sure that the review will be of the quality the evaluator requires. Requirements for agents in improvising scenarios as discussed by [9] also hold for dockets: Their shared goals are weakly defined, leaving much space for creativity, and the constituents (agents) have to work together closely.

7.3 Speech-Act-Based Cooperation

As pointed out earlier, due to performers' autonomy, docket design requires an agreement on the conditions of the collaboration. One approach to model such negotiations is based on the speech act theory, which has already been applied successfully to workflow modeling [31]. Implementations of speech act models are the Action Workflow model [21] and the Business Conversations [17]. For Business Conversations an agent framework [20] has been developed which stresses the autonomy of the actors.

A recent draft of the Internet Engineering Task Force discusses the issue of finding an agreement for data format to be used in communications [13].

7.4 Annotation Models

The structural requirements of document models have already been discussed in section 3.2. An annotation model dealing with these requirements is currently under investigation by [4]. Two specific kinds of annotations are of particular interest to dockets: seals of approval (SOAPs) and usage indicators [26]. SOAPs are used by experts to give ratings to documents. Reviewing with dockets based on enumeration type properties is similar to SOAP assignment. Usage indicators measure a document's value by showing how often it has been read and possibly by whom. Thus, they are similar in purpose as traces produced by dockets.

8 Concluding Remarks

In this paper we give a first presentation of dockets as a conceptual model. We emphasize the potential of capturing the essence of routine processes by generic docket templates and stress the importance of docket design and presentation.

Besides extending the functionality of our docket framework we are particularly interested in further developing our library of docket templates. Thus, we intend to capture in a single conceptual model generic patterns essential for content management and content-based service development.

Our previous work in highly generic computational models with persistence and migration support [16, 18, 19] turns out to be of particular importance for the development of generic and high-level docket technology.

Acknowledgements

Special credit is given to Higher-Order GmbH and some of its customers for numerous discussions of conceptual and technical issues in content management.

Furthermore, we want to thank our colleagues and fellow researchers of the WEL project (Grant by Hamburg State) and the members of the EU-Canada collaboration on Cooperative Information Systems (Grant by Esprit under project ISC-CAN-080 CIS), in particular to John Mylopoulos, Matthias Jarke, and Eric Yu. Credit is also given to Misha Missikoff, Bernd Thalheim and Florian Matthes for continuous constructive contributions to our research.

Finally, special thanks go to our friends, Diane and John Smith from Xerox Corp., for their very valuable input from the perspective of professional document management and presentation.

References

1. Biron, P.V., Malhotra, A. (eds.): XML Schema Part 2: Datatypes. World Wide Web Consortium Working Draft (1999) `http://www.w3.org/TR/xmlschema-2`
2. Brodie, M.L., Mylopoulos, J., Schmidt, J.W. (eds.): On Conceptual Modeling. Springer (1984)
3. Buco, M.J., Chaar, J.K.: On the Integration of Project, Workflow, and Document Management. Technical Report, IBM (1996)
4. Buneman, P., Deutsch, A., Tan, W.: A Deterministic Model for Semistructured Data. Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats (1998)
5. Communications of the ACM, Special Issue on Traceability. **12** (1998)
6. Curran, T., Keller, G., Ladd, A.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model. Prentice Hall (1998)
7. Deacon, T.W.: The Symbolic Species. W.W. Norton & Co. (1997)
8. Grosso P., Veillard, D. (eds.): XML Fragment Interchange. W3C Working Draft (1999) `http://www.w3.org/TR/WD-xml-fragment`
9. Hayes-Roth, B., Brownston, L.: Multi-Agent Collaboration in Directed Improvisation. Technical Report, Stanford University (1994)

10. Hollingsworth, D.: The Workflow Reference Model. Workflow Management Coalition (1995) http://www.aiim.org/wfmc/standards/docs/tc003v11.pdf
11. Jelliffe, R., Jelliffe, R.A.: The XML and SGML Cookbook: Recipes for Structured Information. Prentice Hall (1998)
12. Joeris, G.: Change Management Needs Integrated Process and Configuration Management. ESEC/SIGSOFT FSE (1997) 125-141
13. Klyne, G.: Protocol-Independent Content Negotiation Framework. Draft, Internet Engineering Task Force http://info.internet.isi.edu:80/0/in-drafts/files/draft-ietf-conneg-requirements-02.txt
14. Koksal, P., Arpinar, S.N., Dogac, A.: Workflow History Management. SIGMOD Record **1** (1998) 67-75
15. Maler, E., DeRose, S. (eds.): XML Linking Language (XLink) W3C Working Draft (1998) http://www.w3.org/TR/WD-xlink
16. Mathiske, B., Matthes, F., Schmidt, J.W.: On Migrating Threads. NGITS 1995
17. Matthes, F.: Business Conversations: A High-level System Model for Agent Coordination. DBPL 1997: 355-372
18. Matthes, F., Schmidt, J.W.: Persistent Threads. VLDB 1994: 403-414
19. Matthes, F., Schröder, G., Schmidt, J.W.: Tycoon: A Scalable and Interoperable Persistent System Environment. In: M.P. Atkinson (ed.): Fully Integrated Data Environments. Springer-Verlag (1999, in press)
20. Matthes, F., Wegner, H., Hupe, P.: A Process-Oriented Approach to Software Component Definition. CAiSE 1999: 26-40
21. Medina-Mora, R., Winograd, T., Flores, R., Flores, F.: The ActionWorkflow Approach to Workflow Management Technology. CSCW 1992
22. Muth, P., Weissenfels, J., Gillmann, M., Weikum, G: Workflow History Management in Virtual Enterprises Using a Light-Weight Workflow Management System. RIDE 1999: 148-155
23. Palmer, J.D.: Traceability. In: Dorfman M., Thayer, R. H. (eds.): Software Engineering. IEEE Computer Society Press (1997)
24. Prinz, W., Kolvenbach, S.: Support for Workflows in a Ministerial Environment. CSCW 1996: 199-208
25. Ramesh B., Jarke M.: Towards Reference Models for Requirements Traceability. To appear in: TSE (1999)
26. Röscheisen, M., Mogensen, C., Winograd, T.: Shared Web Annotations As A Platform for Third-Party Value-Added Information Providers: Architecture, Protocols, and Usage Examples. Technical Report (1994) http://www-diglib.stanford.edu/diglib/pub/reports/commentor.html
27. Röscheisen, M., Winograd, T., Paepcke A.: Content Ratings and Other Third-Party Value-Added Information – Defining an Enabling Platform. D-Lib Magazine (1995) http://mirrored.ukoln.ac.uk/lis-journals/dlib/dlib/dlib/august95/stanford/08roscheisen.html
28. Ryu, Y.U.: Specification of Contractual Obligations in Formal Business Communication. DKE 26(3): 309-326 (1998)
29. Schmidt, J.W., Schröder, G., Niederée, C., Matthes, F.: Linguistic and Architectural Requirements for Personalized Digital Libraries. Int. J. Digit. Libr. **1** (1997) 89-104
30. Warburg Electronic Library, http://www.sts.tu-harburg.de/projects/WEL
31. Winograd, T.: A Language/Action Perspective on the Design of Cooperative Work. Human-Computer Interaction **1** (1987-88) 3-30
32. Yu, E.S.K., Mylopoulos, J.: From E-R to "A-R" – Modelling Strategic Actor Relationships for Business Process Reengineering. ER 1994: 548-565