A Deontic Logic for Group-Oriented Web Information Systems

Klaus-Dieter Schewe[1] Bernhard Thalheim[2] Roland Kaschek[1]

[1] Massey University, Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand
[k.d.schewe|r.h.kaschek]@massey.ac.nz

[2] Christian Albrechts University Kiel
Department of Computer Science and Applied Mathematics
Olshausenstr. 40, D-24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Abstract: Group-oriented web information systems (GWISs) are web-based data-intensive systems that are used by a group of people in order to fulfill common tasks. In particular, there are different roles for the users, each associated with some rights and obligations. The group orientation implies that some of these obligations and rights depend on actions done by other roles. In this article a deontic logic for formulating and reasoning about such obligations and rights is presented.

1 Introduction

A *web information system* (WIS) is a data-intensive information system that is realized and distributed over the web with user access via web browsers. Information is made available via pages including a navigation structure between them and to sites outside the system. Furthermore, there should also be operations to retrieve data from the system or to update the underlying database(s).

Such WISs have been applied to various areas including e-business and e-commerce, e-learning, information services, cooperative web-based systems, but neither is there a commonly agreed methodology for all these areas nor an understanding of the particular needs for some applications. Instead of this, a lot of work is concentrated on a problem triplet consisting of content, navigation and presentation, i.e. modelling databases, hypertext structures and page layout. Examples of such development methodologies are among many others the ARANEUS framework [AGS98, BGP00], the OOHDM framework [RGS00, RSL99, SR98], the the WebML work in [CFP99, CFB+03, FP98] and the Co-Design Approach [DT01, KSWM04, STZ04]. In addition, a lot of work has been investigated on appropriate presentations [GT99, VLH02].

Our own work emphasises a methodology oriented at abstraction layers and the co-design

of structure, operations and interfaces. At a high level of abstraction this first leads to *storyboarding*, an activity that addresses the design of underlying application stories. Furtheron, the scenes in the stories have to be adequately supported, which leads to *media types*, i.e. views extended by adaptivity and hierarchies.

In this article we concentrate on group-oriented WISs (GWISs). A GWIS is used by a group of users in fulfillment of common tasks. In particular, there are different roles for the users, each associated with some rights and obligations. The group orientation implies that some of these obligations and rights depend on actions done by other roles. We will define a deontic logic for formulating and reasoning about such obligations and rights following [BWM02, DMWK96, ES99, WM93, WM91]. The logic will direct the navigation of a user in a particular role through the system by forbidding certain actions, unless some deontic pre-conditions are satidfied. Furthermore, the logic permits deciding whether goals of group-oriented tasks can be reached or not.

The deontic logic will be linked to processes that describe actions in the story space. The processes are build from the actions at the scenes in the story space, and they will form a many-sorted Kleene algebra with tests [Ko97, ST04].

In Section 2 we describe how the story space can be modelled. Starting from a simple description by a labelled multi-graph we derive a process algebra along the lines of the language `SiteLang` [DT01]. We will illustrate the language with an example from an e-banking application. In Section 3 we introduce roles, which basically lead to rights and obligations. We then formalize them using a deontic logic, and continue the e-banking application. Finally, we conclude with a brief summary.

2 Storyboarding

In order to fullfil tasks users navigate between abstract locations, and on this navigation path they execute a number of actions. We regard a location together with local actions, i.e. actions that do not change the location, as a unit called *scene*. Then a WIS can be decribed by a edge-labelled directed multi-graph, in which the vertices represent the scenes, and the edges represent transitions between scenes. Each such transition may be labelled by an action executed by the user. If such a label is missing, the transition is due to a simple navigation link. The whole multi-graph is then called the *story space*.

Roughly speaking, a *story* is a path in the story space. It tells what a user of a particular type might do with the system.

The combination of different stories to a subgraph of the story space can be used to describe a “typical” use of the WIS for a particular task. Therefore, we call such a subgraph a *scenario*. Usually storyboarding starts with modelling scenarios instead of stories, coupled by the integration of stories to the story space.

At a finer level of details we may add a triggering *event*, a *precondition* and a *postcondition* to each action, i.e. we specify exactly, under which conditions an action can be executed and which effects it will have.

Looking at scenarios or the whole story space from a different angle, we may concentrate on the flow of actions:

- For the purpose of storyboarding actions can be treated as being atomic, i.e. we are not yet interested in how an underlying database might be updated. Then each action also belongs to a uniquely determined scene.
- Actions have pre- and postconditions, so we can use annotations to express conditions that must hold before or after an action is executed.
- Actions can be executed sequentially or parallel, and we must allow (demonic) choice between actions.
- Actions can be iterated.
- By adding an action with no effect we can then also express optionality and iteration with at least one execution.

These possibilities to combine actions lead to operators of an algebra, which we will call a *story algebra*. Thus, we can describe a story space by an element of a suitable story algebra. We should, however, note already that story algebras have to be defined as being many-sorted in order to capture the association of actions with scenes.

In order to describe scenarios (or the whole story space) we use the language `SiteLang` from [DT01]. This languages uses standard process algebra constructors for sequences, parallel execution, choice, iteration as well as guards and post-guards.

Formally, let $\mathcal{S} = \{s_1, \ldots, s_n\}$ be a set of scenes, and let $\mathcal{A} = \{\alpha_1, \ldots, \alpha_k\}$ be a set of (atomic) actions. Furthermore, assume a mapping $\sigma : \mathcal{A} \to \mathcal{S}$, i.e. with each action $\alpha \in \mathcal{A}$ we associate a scene $\sigma(\alpha)$. Then define inductively the set of *processes* $\mathcal{P} = \mathcal{P}(\mathcal{A}, \mathcal{S})$ determined by $\mathcal{A}$ and $\mathcal{S}$. Furthermore, we can extend σ to a partial mapping $\mathcal{P} \to \mathcal{S}$. We use notation that highlights the relationship to Kleene algebras with tests:

- Each action $\alpha \in \mathcal{A}$ is also a process, i.e. $\alpha \in \mathcal{P}$, and the associated scene $\sigma(\alpha)$ is already given.
- 1 and 0 are processes, for which σ is undefined. 1 is a process with no effect, while 0 is not executable. These processes are usually called `skip` and `fail`, respectively.
- If p_1 and p_2 are processes, then also the *sequence* $p_1 \cdot p_2$ is a process. Furthermore, if $\sigma(p_1) = \sigma(p_2) = s$ or one of the p_i is 1, then $\sigma(p_1 \cdot p_2)$ is also defined and equals s, otherwise it is undefined. We take the freedom to write also $p_1 p_2$ instead of $p_1 \cdot p_2$.
- If p_1 and p_2 are processes, then also the *parallel process* $p_1 \| p_2$ is a process. Furthermore, if $\sigma(p_1) = \sigma(p_2) = s$ or one of the p_i is 1, then $\sigma(p_1 \| p_2)$ is also defined and equals s, otherwise it is undefined. However, for the purpose of reasoning about storyboarding it is advantageous to use $p_1 \cdot p_2$ instead of $p_1 \| p_2$ and to assume commutativity, i.e. $p_1 p_2 = p_2 p_1$ in this case.

- If p_1 and p_2 are processes, then also the *choice* $p_1 + p_2$ is a process. Furthermore, if $\sigma(p_1) = \sigma(p_2) = s$ or one of the p_i is 1, then $\sigma(p_1 + p_2)$ is also defined and equals s, otherwise it is undefined.
- If p is a process, then also the *iteration* p^* is a process with $\sigma(p^*) = \sigma(p)$, if $\sigma(p)$ is defined.
- If p is a process and φ is a boolean condition, then the *guarded process* $\varphi \cdot p$ and the *post-guarded process* $p \cdot \varphi$ are processes with $\sigma(\varphi p) = \sigma(p\varphi) = \sigma(p)$, if $\sigma(p)$ is defined.
- In addition, we use $\varphi \cdot \psi$ to denote logical conjunction, $\varphi + \psi$ to denote disjunction, and $\bar{\varphi}$ to denote negation. Then 1 and 0 stand for true and false, respectively.

Note that the overloaded use of $+$ for disjunction and choice, of $\cdot$ for conjunction and sequence, of 1 for true and `skip`, and of 0 for false and `fail` does not cause problems [Ko97]. In [ST04] it was shown that this definition of a process algebra defines indeed a many-sorted Kleene algebra with tests, where the sorts are the scenes.

The following process describes activities in on-line loan applications. Basically, we have a part dealing with the application for two types of loans, home loans and mortgages, and a part dealing with their assessment.

(enter_loan_system
((φ_0 look_at_loans_at_a_glance +
(φ_1 request_home_loan_details
(look_at_home_loan_samples + 1) φ_3) +
(φ_2 request_mortgage_details
(look_at_mortgage_samples + `skip`) φ_4))* φ_5)
(select_home_loan $\{\varphi_6\}$ + select_mortgage $\{\varphi_7\}$)
((φ_6 (provide_applicant_details
(provide_applicant_details + `skip`)
(describe_loan_purpose enter_amount_requested
enter_income_details)
select_hl_terms_and_conditions) φ_8) +
(φ_7 (provide_applicant_details provide_applicant_details*
(describe_object enter_mortgage_amount
describe_securities*)
(enter_income_details enter_obligations*)
(($\bar{\varphi}_{12}$ select_m_terms_and_conditions
calculate_payments)*
φ_{12} select_m_terms_and_conditions)) φ_9))
confirm_application $(\varphi_{10} + \varphi_{11})$) +
(enter_loan_assessment
((select_hl_application asess_hl_application $(\varphi_{13} + \varphi_{14})$
(φ_{13} ((reject_hl_application φ_{15}) +
(($(\bar{\varphi}_{14}\bar{\varphi}_{15})$ explore_hl_alternative

(modify_hl_application + 1))* $(\varphi_{14} + \varphi_{15})$))) +
(φ_{14} ((accept_hl_application φ_{16}) +
(explore_hl_alternative
(modify_hl_application + 1)* φ_{16})))) +
(select_m_application assess_m_application $(\varphi_{17} + \varphi_{18})$
((φ_{17} arrange_meeting $(\varphi_{19} + \varphi_{20})$) +
(φ_{18} accept_m_application φ_{20})))))

involving the conditions

$\varphi_0 \equiv$ information_about_loan_types_needed
$\varphi_1 \equiv$ information_about_home_loans_needed
$\varphi_2 \equiv$ information_about_mortgages_needed
$\varphi_3 \equiv$ home_loans_known
$\varphi_4 \equiv$ mortgages_known
$\varphi_5 \equiv$ available_loans_known
$\varphi_6 \equiv$ home_loan_selected
$\varphi_7 \equiv$ mortgage_selected
$\varphi_8 \equiv$ home_loan_application_completed
$\varphi_9 \equiv$ mortgage_application_completed
$\varphi_{10} \equiv$ applied_for_home_loan
$\varphi_{11} \equiv$ applied_for_mortgage
$\varphi_{12} \equiv$ payment_options_clear
$\varphi_{13} \equiv$ hl_critical
$\varphi_{14} \equiv$ hl_acceptable
$\varphi_{15} \equiv$ hl_rejected
$\varphi_{16} \equiv$ hl_accepted
$\varphi_{17} \equiv$ m_critical
$\varphi_{18} \equiv$ m_acceptable
$\varphi_{19} \equiv$ mortgage_rejected
$\varphi_{20} \equiv$ mortgage_accepted

Using

$\alpha_1 =$ enter_loan_system $\alpha_2 =$ look_at_loans_at_a_glance
$\alpha_3 =$ request_home_loan_details $\alpha_4 =$ request_mortgage_details
$\alpha_5 =$ look_at_home_loan_samples $\alpha_6 =$ look_at_mortgage_samples
$\alpha_7 =$ select_home_loan $\alpha_8 =$ provide_applicant_details
$\alpha_9 =$ describe_loan_purpose $\alpha_{10} =$ enter_amount_requested
$\alpha_{11} =$ enter_income_details $\alpha_{12} =$ select_hl_terms_and_conditions
$\alpha_{13} =$ select_mortgage $\alpha_{14} =$ describe_object
$\alpha_{15} =$ enter_mortgage_amount $\alpha_{16} =$ describe_securities
$\alpha_{17} =$ enter_obligations $\alpha_{18} =$ select_m_terms_and_conditions
$\alpha_{19} =$ calculate_payments $\alpha_{20} =$ confirm_application
$\alpha_{21} =$ enter_loan_assessment $\alpha_{22} =$ select_hl_application
$\alpha_{23} =$ assess_hl_application $\alpha_{24} =$ reject_hl_application
$\alpha_{25} =$ explore_hl_alternative $\alpha_{26} =$ modify_hl_application
$\alpha_{27} =$ accept_hl_application $\alpha_{28} =$ select_m_application
$\alpha_{29} =$ assess_m_application $\alpha_{30} =$ arrange_meeting
$\alpha_{31} =$ accept_m_application

we can rewrite the story space by the process expression

$$(\alpha_1((\varphi_0\alpha_2 + \varphi_1\alpha_3(\alpha_5+1)\varphi_3 + \varphi_2\alpha_4(\alpha_6+1)\varphi_4)^*\varphi_5)(\alpha_7\varphi_6 + \alpha_{13}\varphi_7)$$
$$(\varphi_6\alpha_8(\alpha_8+1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8 + \varphi_7\alpha_8\alpha_8^*\alpha_{14}\alpha_{15}\alpha_{16}^*\alpha_{11}\alpha_{17}(\overline{\varphi_{12}}\alpha_{18}\alpha_{19})^*\varphi_{12}\alpha_{18}\varphi_9)$$
$$\alpha_{20}(\varphi_{10} + \varphi_{11}))+$$
$$(\alpha_{21}((\alpha_{22}\alpha_{23}(\varphi_{13} + \varphi_{14})(\varphi_{13}((\alpha_{24}\varphi_{15}) + (((\overline{\varphi_{14}} + \overline{\varphi_{15}})\alpha_{25}(\alpha_{26}+1))^*(\varphi_{14} + \varphi_{15}))))$$
$$+(\varphi_{14}((\alpha_{27}\varphi_{16}) + (\alpha_{25}(\alpha_{26}+1)^*\varphi_{16}))))+$$
$$(\alpha_{28}\alpha_{29}(\varphi_{17} + \varphi_{18})((\varphi_{17}\alpha_{30}(\varphi_{19} + \varphi_{20})) + (\varphi_{18}\alpha_{31}\varphi_{20})))))$$

together with the constraints

$$\alpha_9\alpha_{10} = \alpha_{10}\alpha_9, \quad \alpha_9\alpha_{11} = \alpha_{11}\alpha_9, \quad \alpha_{11}\alpha_{10} = \alpha_{10}\alpha_{11},$$
$$\alpha_{14}\alpha_{15} = \alpha_{15}\alpha_{14}, \quad \alpha_{14}\alpha_{16} = \alpha_{16}\alpha_{14}, \quad \alpha_{16}\alpha_{15} = \alpha_{15}\alpha_{16}, \quad \alpha_{11}\alpha_{17} = \alpha_{17}\alpha_{11},$$
$$\varphi_5\varphi_0 = 0, \quad \varphi_5\varphi_1 = 0, \quad \varphi_5\varphi_2 = 0, \quad \varphi_{10}\varphi_{11} = 0, \quad \varphi_6\varphi_7 = 0$$

In [ST04] it has been shown how to exploit equational reasoning with Kleene algebras with test to personalise the story space.

3 Obligations and Rights Associated with Roles

The presence of roles indicates a particular purpose of the system. For instance, in an on-line loan systems we may wish to distinguish between *customers* who apply for home loans and mortgages and *bank clerks* who assess these applications, ask for modifications or suggest them, and decide on acceptance or rejection of applications. For instance, in the simplified story space in the previous section we have a clear distinction between the alternative starting with $\alpha_1 =$ enter_loan_system, which indicates the application part of the system used by customers, and the alternative starting with $\alpha_{21} =$ enter_loan_assessment, which indicates the assessment part used by bank clerks.

A *role* is defined by the set of actions that a user with this role may execute. Thus, we first associate with each scene in the story space a set of role names, i.e. whenever a user comes across a particular scene, s/he will have to have one of these roles. Furthermore, a role is usually associated with obligations and rights, i.e. which actions have to be executed or which scenes are disclosed.

An *obligation* specifies what a user in a particular role has to do. A *right* specifies what a user in a particular role is permitted to do. Both obligations and rights together lead to complex deontic integrity constraints. We use the following logical language $\mathcal{L}$ adopted from [ES99] for this purpose:

- All propositional atoms used in the story space are also atoms of $\mathcal{L}$.
- If α is an action on scene s and r is a role associated with s, then $\mathbf{O}\ do(r, \alpha)$ is an atom of $\mathcal{L}$.

- If α is an action on scene s and r is a role associated with s, then $\mathbf{P}\ do(r, \alpha)$ is an atom of $\mathcal{L}$.
- If α is an action on scene s and r is a role associated with s, then $\mathbf{F}\ do(r, \alpha)$ is an atom of $\mathcal{L}$.
- If α is an action on scene s and r is a role associated with s, then $do(r, \alpha)$ is an atom of $\mathcal{L}$.
- For $\varphi, \psi \in \mathcal{L}$ we also have that $\bar{\varphi}$, $\varphi \cdot \psi$, $\varphi + \psi$ are formulae in $\mathcal{L}$. Then we use the usual shortcuts $\varphi \rightarrow \psi$ for $\bar{\varphi} + \psi$, and $\varphi \leftrightarrow \psi$ for $(\bar{\varphi} + \psi)(\varphi + \bar{\psi})$.

Informally, $\mathbf{O}\ do(r, \alpha)$ means that a user with role r is obliged to perform action α. $\mathbf{P}\ do(r, \alpha)$ means that a user with role r is permitted to perform action α, and $\mathbf{F}\ do(r, \alpha)$ means that a user with role r is forbidden to perform action α. Furthermore, $do(r, \alpha)$ means that a user in role r actually does execute α.

We use this logic to formulate deontic constraints on the story space. Then a WIS turns into a group-oriented WIS, if there are constraints that link actions by one role to actions by another role, i.e. a global goal can only be achieved, if a group of users in different roles collaborate. In this way, complex business rules can be expressed as deontic constraints to the story space.

In the on-line loan application example that we used in the previous section we may identify several deontic constraints. For instance,

$$\varphi_6 + \varphi_7 \rightarrow \mathbf{O}\ do(\text{customer}, \alpha_8)$$

specifies that a customer who has selected a home loan or a mortgage is obliged to provide personal details of the applicant(s).

$$\varphi_7 \rightarrow \mathbf{O}\ do(\text{customer}, \alpha_{16}) \cdot \mathbf{O}\ do(\text{customer}, \alpha_{17})$$

expresses that if a mortgage is selected, a customer has to provide securities and obligations.

$$\varphi_{10} + \varphi_{11} \rightarrow \mathbf{O}\ do(\text{clerk}, \alpha_{21})$$

specifies that if a home loan or mortgage has been applied for, a bank clerk will have to enter the loan assessment part of the system. This links customers to bank clerks, in particular, as φ_{10} corresponds to a completed application for a home loan by a customer, and φ_{11} corresponds to a completed application for a mortgage. If these were simple actions, say α and β, respectively, they could have been expressed by $\varphi_{10} \leftrightarrow do(\text{customer}, \alpha)$ and $\varphi_{11} \leftrightarrow do(\text{customer}, \beta)$, but we only permitted simple actions in the deontic atoms.

$$\varphi_{13} \rightarrow \mathbf{P}\ do(\text{clerk}, \alpha_{24}) \cdot \mathbf{P}\ do(\text{clerk}, \alpha_{26})$$

expresses that if the assessment of a home loan shows that the application is critical, the bank clerk is permitted to reject the application. S/he is also permitted to modify the application. According to the story space, only one of these activities will be taken. Similarly,

$$\varphi_{14} \rightarrow \mathbf{F}\ do(\text{clerk}, \alpha_{24}) \cdot \mathbf{P}\ do(\text{clerk}, \alpha_{26})$$

specifies that if the assessment of a home loan shows that the application is acceptable, the bank clerk is not permitted to reject the application. However, s/he may still modify the application, e.g. if this turns out to be advantageous for the customer.

$$\varphi_{18} \rightarrow \ do(\text{clerk}, \alpha_{31})$$

and

$$\varphi_{17} \rightarrow \ do(\text{clerk}, \alpha_{30})$$

express that in case a mortgage application is acceptable, the bank clerk will accept it, whereas in case it turns out to be critical the bank clerk will arrange a meeting to discuss the problems and possible options.

$$do(\text{clerk}, \alpha_{30}) \cdot \overline{do(\text{customer}, \alpha_1)} \rightarrow \varphi_{19}$$

specifies that if a bank clerk arranges a meeting to discuss mortgage options, but the customer does not enter the loan system for a revised application, the mortgage application will be rejected.

Let us conclude this section with a brief discussion of the semantics of our logic, for which we use status sets. A *status set* of $\mathcal{L}$ is simply a set $\mathcal{S}$ of atoms satisfying

$$\mathbf{P}\ do(r, \alpha) \in \mathcal{S} \quad \text{iff} \quad \mathbf{F}\ do(r, \alpha) \notin \mathcal{S}$$

If Σ is a set of constraints described as formulae in $\mathcal{L}$, then we say that the status set $\mathcal{S}$ is *feasible* iff it satisfies Σ in the usual propositional sense, i.e. $\mathcal{S}$ determines the truth values of the atoms, and the usual interpretation for negation, conjunction and disjunction applies. Furthermore, we call $\mathcal{S}$ *closed* iff we have

$$\begin{aligned} \mathbf{O}\ do(r, \alpha) \in \mathcal{S} \quad &\Rightarrow \quad \mathbf{P}\ do(r, \alpha) \in \mathcal{S} \\ \mathbf{O}\ do(r, \alpha) \in \mathcal{S} \quad &\Rightarrow \quad do(r, \alpha) \in \mathcal{S} \\ do(r, \alpha) \in \mathcal{S} \quad &\Rightarrow \quad \mathbf{P}\ do(r, \alpha) \in \mathcal{S} \end{aligned}$$

Obviously, we can use these rules to build the closure of any status set. A sequence $\mathcal{S}_0 \rightarrow \mathcal{S}_1 \rightarrow \cdots \rightarrow \mathcal{S}_n$ is *enabled* iff all $\mathcal{S}_i$ are closed and feasible with respect to Σ, and for each $i = 0, \ldots, n-1$ there is some action α_i and some role r with $do(r, \alpha_i) \in \mathcal{S}_i$ such that the excution of α_i will produce the status set $\mathcal{S}_{i+1}$ and the sequence of actions $\alpha_0, \alpha_1, \ldots, \alpha_n$ is permitted by the story space.

In [WM91, WM93] it has been shown how deontic logic can be modelled by multi-modal logics which allows the rules for modal logics to be applied to our deontic logic.

4 Conclusion

In this article we approached the specification of obligations and rights of roles in web information systems. On the basis of an algebraic story space specification, which in fact

leads to a many-sorted Kleene algebra with tests, we introduced a deontic action logic. In a nutshell this logic allow to specify conditions, under which actions by a particular role are permitted, forbidden or obliged, and conditions for their actal execution. A WIS turns into a group-oriented WIS, if there are constraints that link actions by one role to actions by another role. The deontic logic will direct the navigation of a user in a particular role through the system by forbidding certain actions, unless some deontic pre-conditions are satidfied. Furthermore, the logic permits deciding whether goals of group-oriented tasks can be reached or not. This is achieved by using a semantics based on sequences of feasible status sets.

This formalisation of story spaces and deontic constraints can be used in two ways to reason about the specification of group-oriented web information systems. The fact that we obtain a many-sorted Kleene algebra with tests can be used for equational reasoning. One application of equational reasoning is the personalisation of the story space, which basically means to simplify the algebraic story space expression according to preference rules and intentions that are modelled by postconditions. This approach has been described in [ST04]. In addition, we can reason about the deontic logic, which can tell us, whether a certain common goal can be achieved or not.

So far, our approach to reasoning about story boards in purely propositional. Thus, the interesting challenging question is what can be achieved by combining the deontic logic developed in this article with the specification of media types that support the individual scenes [ST04]. Fortunately, if we restrict our attention to a clausal-form logic as in [ES99], some results are still possible. However, the general approach would also require that we switch from Kleene algebras with test to general dynamic logic [HKT00]. These two problems constitute major areas of future research.

References

[AGS98] Atzeni, P., Gupta, A., und Sarawagi, S.: Design and maintenance of data-intensive web-sites. In: *Proceeding EDBT'98*. volume 1377 of *LNCS*. pp. 436–450. Springer-Verlag. Berlin. 1998.

[BGP00] Baresi, L., Garzotto, F., und Paolini, P.: From web sites to web applications: New issues for conceptual modeling. In: *ER Workshops 2000*. volume 1921 of *LNCS*. pp. 89–100. Springer-Verlag. Berlin. 2000.

[BWM02] Broersen, J., Wieringa, R., und Meyer, J.-J. C.: A fixed-point characterization of a deontic logic of regular action. *Fundamenta Informaticae*. 49(4):107–128. 2002.

[CFB+03] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., und Matera, M.: *Designing Data-Intensive Web Applications*. Morgan Kaufmann. San Francisco. 2003.

[CFP99] Ceri, S., Fraternali, P., und Paraboschi, S.: Data-driven, one-to-one web site generation for data-intensive applications. In: *Proceedings of the Conference on Very Large Databases (VLDB 1999)*. pp. 615–626. IEEE Computer Society. 1999.

[DMWK96] Dignum, F., Meyer, J.-J. C., Wieringa, R., und Kuiper, R.: A modal approach to intentions, commitments and obligations: Intention plus commitment yields obligation. In: *DEON 1996*. pp. 80–97. 1996.

[DT01] Düsterhöft, A. und Thalheim, B.: SiteLang: Conceptual modeling of internet sites. In: Kunii, H. S., Jajodia, S., und Sølvberg, A. (Eds.), *Conceptual Modeling – ER 2001*. volume 2224 of *LNCS*. pp. 179–192. Springer-Verlag. Berlin. 2001.

[ES99] Eiter, T. und Subrahmanian, V. S.: Deontic action programs. In: Polle, T., Ripke, T., und Schewe, K.-D. (Eds.), *Fundamentals of Information Systems*. pp. 37–54. Kluwer Academic Publishers. 1999.

[FP98] Fraternali, P. und Paolini, P.: A conceptual model and a tool environment for developing more scalable, dynamic and customizable web applications. In: *Proceedings EDBT'98*. volume 1377 of *LNCS*. pp. 422–435. Springer-Verlag. 1998.

[GT99] Gehrke, D. und Turban, E.: Determinants of successful website design: Relative importance and recommendations for effectiveness. In: *Proceedings HICSS'99*. Springer-Verlag. 1999.

[HKT00] Harel, D., Kozen, D., und Tiuryn, J.: *Dynamic Logic*. The MIT Press. Cambridge (MA), USA. 2000.

[Ko97] Kozen, D.: Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*. 19(3):427–443. 1997.

[KSWM04] Kaschek, R., Schewe, K.-D., Wallace, C., und Matthews, C.: Story boarding for web information systems. In: Taniar, D. und Rahayu, W. (Eds.), *Web Information Systems*. pp. 1–33. IDEA Group. 2004.

[RGS00] Rossi, G., Garrido, A., und Schwabe, D.: Navigating between objects: Lessons from an object-oriented framework perspective. *ACM Computing Surveys*. 32(1). 2000.

[RSL99] Rossi, G., Schwabe, D., und Lyardet, F.: Web application models are more than conceptual models. In: Chen, P. P.-S. (Ed.), *Advances in Conceptual Modeling*. volume 1727 of *LNCS*. pp. 239–252. Springer-Verlag. Berlin. 1999.

[SR98] Schwabe, D. und Rossi, G.: An object oriented approach to web-based application design. *TAPOS*. 4(4):207–225. 1998.

[ST04] Schewe, K.-D. und Thalheim, B.: Conceptual modelling of web information systems. Technical Report 3/2004. Massey University, Department of Information Systems. 2004. available from http://infosys.massey.ac.nz/research/rs_techreports.html.

[STZ04] Schewe, K.-D., Thalheim, B., und Zlatkin, S.: Modelling actors and stories in web information systems. In: *Third International Conference on Information Systems Technology and its Applications (ISTA 2004)*. 2004. to appear.

[VLH02] Van Duyne, D. K., Landay, J. A., und Hong, J. I.: *The Design of Sites*. Addison-Wesley. Boston. 2002.

[WM91] Wieringa, R. und Meyer, J.-J. C.: Actor-oriented specification of deontic integrity constraints. In: *Mathematical Fundamentals of Database Systems (MFDBS 1991)*. pp. 89–103. 1991.

[WM93] Wieringa, R. und Meyer, J.-J. C.: Actors, actions, and initiative in normative system specification. *Annals of Mathematics and Artificial Intelligence*. 7(1-4):289–346. 1993.