# Modelling Actors and Stories in Web Information Systems

Klaus-Dieter Schewe[1] Bernhard Thalheim[2] Sergiy Zlatkin[1]

[1] Massey University, Department of Information Systems &
Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand
[k.d.schewe|s.zlatkin]@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:** The design of web information systems (WISs) requires a clear picture of the intended users and their behaviour. WIS users are called *actors* and classified according to their roles, intentions and behaviour. This leads to various propositional and deontic constraints on the *story space*, i.e. the abstract specification of WIS locations and activities associated with them. The knowledge obtained from the modelling of actors can be used to personalise the story space to the needs of a particular actor.

**Keywords:** web information system, storyboarding, user modelling, personalisation

## 1 Introduction

A *web information system* (WIS) is a database-backed 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).

Various approaches to develop design methods for WISs have been proposed so far. Most of them such as the ARANEUS framework [AGS98, BGP00, BCFM00, MMA99], the OOHDM framework [RGS00, RSL99, SR98] and the WebML work in [CFP99, CFB+03, FP98, Fr99] focus on a problem triplet consisting of content, navigation and presentation. This leads to modelling databases, hypertext structures and page layout.

Our own work in [ST01] emphasises a methodology oriented at abstraction layers and the co-design of structure, operations and interfaces. As WIS are open systems in the sense that everyone who has access to the web may turn up as a user, their design requires a clear picture of the intended users and their behaviour. This includes knowledge about the used access channels and end-devices. At a high level of abstraction this first leads

*storyboarding*, an activity that addresses the design of an underlying application story [FT99, KSWM04]. As soon as WISs become large, it becomes decisive that such an underlying application story is well designed. Furtheron, the scenes in the stories have to be adequately supported. For this our work focusses on the integration of traditional methods for the design of data-intensive information systems with new methods addressing the challenges arising from the web-presentation and the open access. This leads to *media types*, which cover extended views, adaptivity, hierarchies and presentation style options.

Users of a WIS can be classified according to their roles, intentions and behaviour. We use the term *actor* for such a group of users. In this article, we emphasize the modelling of actors. The *role* of an actor indicates a particular purpose of the system. As such it is usually associated with obligations and rights, which lead to deontic and dynamic integrity constraints. Roles are also connected with the *tasks*, but tasks will be handled separately. The *intention* of an actor can be modelled by goals, i.e. postconditions to the story space. Modelling the behaviour of an actor further leads to *user profiles*, which can be modelled by giving values for various properties that characterize a user. Furthermore, each profile leads to rules that can again be expressed by constraints on the story space.

The core of a story space itself can be expressed by a directed multi-graph, in which the vertices represent scenes and the edges actions by the user including navigation. If more details are added, application stories can be expressed by some form of process algebra. In this article we also emphasize the modelling of the story space. Furthermore, we show how the knowledge obtained from the modelling of actors can be used to personalise the story space to the needs of a particular actor.

## 2 Modelling Stories

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. Further extensions to scenes such as adaptivity, presentation, tasks and roles have been discussed in [BKST04] and [FKST00].

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 `skip` 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.

**Example 1.** Consider the loan application from [SKWM03]. A rough sketch of the story space (used also in [ST04]) can be described as follows:

enter_loan_system ;
( ( $\{\varphi_0\}$ look_at_loans_at_a_glance □
( $\{\varphi_1\}$ request_home_loan_details ;
( look_at_home_loan_samples □ `skip` ) $\{\varphi_3\}$ ) □
( $\{\varphi_2\}$ request_mortgage_details ;
( look_at_mortgage_samples □ `skip` ) $\{\varphi_4\}$ ) )$^*$ $\{\varphi_5\}$ ) ;
( select_home_loan $\{\varphi_6\}$ □ select_mortgage $\{\varphi_7\}$ ) ;
( ( $\{\varphi_6\}$ ( provide_applicant_details ;
( provide_applicant_details □ `skip` ) ;
( describe_loan_purpose ‖ enter_amount_requested ‖
enter_income_details ) ;
select_hl_terms_and_conditions ) $\{\varphi_8\}$ ) □
( $\{\varphi_7\}$ ( provide_applicant_details ; provide_applicant_details$^*$ ;
( describe_object ‖ enter_mortgage_amount ‖
describe_securities$^*$ ) ;
( enter_income_details ‖ enter_obligations$^*$ ) ;
( ( $\{\neg\varphi_{12}\}$ select_m_terms_and_conditions ;

calculate_payments )$^*$ ;
$\{\varphi_{12}\}$ select_m_terms_and_conditions ) ) $\{\varphi_9\}$ ) ) ;
confirm_application $\{\varphi_{10} \vee \varphi_{11}\}$

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
$\varphi_8 \equiv$ home_loan_application_completed
$\varphi_9 \equiv$ mortgage_application_completed
$\varphi_{10} \equiv$ applied_for_home_loan
$\varphi_{12} \equiv$ payment_options_clear
$\varphi_3 \equiv$ 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
$\varphi_{11} \equiv$ applied_for_mortgage

The work in [ST04] contains a mathematical formalisation of story algebras using Kleene algebras with tests [Ko97].

## 3 Modelling Actors: Roles, Obligations, Rights and Intentions

The presence of roles indicates a particular purpose of the system. For instance, in a web-based conference system we may have roles for the programme committee chair(s), the programme committee members, and for authors. On the other hand, in an on-line loan systems we may not wish to distinguish roles, as all actors will only appear in the one role of a customer.

A *role* is defined by the set of actions that an actor with this role may execute. Thus, we first associate with each scene in the story space a set of role names, i.e. whenever an actor 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 an actor in a particular role has to do. A *right* specifies what an actor 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 $\mathcal{L}$ for this purpose:

- All propositional atoms are also atoms of $\mathcal{L}$.
- If $\alpha$ 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 $\alpha$ 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 $\alpha$ is an action on scene $s$ and $r$ is a role associated with $s$, then $\mathbf{F}\ do(r, \alpha)$ is an atom of $\mathcal{L}$.
- For $\varphi, \psi \in \mathcal{L}$ we also have $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$ and $\varphi \Leftrightarrow \psi$ are also formulae in $\mathcal{L}$.

The interpretation is standard. In particular, $\mathbf{O}\ do(r, \alpha)$ means that an actor with role $r$ is obliged to perform action $\alpha$, $\mathbf{P}\ do(r, \alpha)$ means that an actor with role $r$ is permitted to perform action $\alpha$, and $\mathbf{F}\ do(r, \alpha)$ means that an actor with role $r$ is forbidden to perform action $\alpha$.

**Example 2.** Take the on-line loan example from [SKWM03]. Though this application only contains one role customer, this customer has the obligation to leave his/her details, once a home loan or a mortgage has been selected. Furthermore, if a mortgage is selected, the customer is obliged to describe securities and to enter obligations, i.e. we obtain the deontic constraints

$$\text{home\_loan\_selected} \vee \text{mortgage\_selected} \Rightarrow \mathbf{O}\ do(\text{customer}, \text{provide\_applicant\_details})$$

and

$$\begin{aligned}\text{mortgage\_selected} \Rightarrow\ & \mathbf{O}\ do(\text{customer}, \text{describe\_securities}) \\ \wedge & \mathbf{O}\ do(\text{customer}, \text{enter\_obligations})\end{aligned}$$

We may of course extend the on-line loan system in a way that the processing of a loan application will be included. In this case we would obtain additional roles, e.g. bank_clerk or mortgage_advisor, and additional deontic constraints.

The *intention* of an actor can be expressed by goals, which can be modelled by postconditions. For instance, in an on-line loan system the goal of a user who is looking for a mortgage leads to the propositional atom applied_for_mortgage.

## 4 User Profiles

Modelling the behaviour of an actor leads to *user profiles*. We may ask which properties characterize a user and provide values for each of these properties. Each combination of such values defines a profile, but usually the behaviour for some of these profiles is the same. Furthermore, each profile leads to preferences that can be expressed by constraints. Preferences can be stated as follows:

1. An equation $p_1 + p_2 = p_1$ expresses an unconditional preference of activity $p_1$ over $p_2$.
2. An equation $\varphi(p_1 + p_2) = \varphi p_1$ expresses an conditional preference of activity (or process) $p_1$ over $p_2$ in case the condition $\varphi$ is satisfied.

3. Similarly, an equation $p(p_1 + p_2) = pp_1$ expresses another conditional preference of activity (or process) $p_1$ over $p_2$ after the activity (or process) $p$.

4. An equation $p_1p_2 + p_2p_1 = p_1p_2$ expresses a preference of order.

The dimensions used in user profiles depend on the application. A rough classification of sources for such dimensions is the following:

- the ability to search for solutions, solve problems, detect and resolve conflicts, schedule work tasks;
- the communication skills and computer literacy;
- the knowledge and education level regarding the task domain;
- the frequency and intensity of system usage;
- the way information is handled, i.e. the direction of the information flow, the necessary and optional input, the intended information usage, the amount and size of information and the complexity of information;
- the experience in working with the system and with associated tasks.

Formally, in order to describe such user profiles, we start with a finite set $\Delta$ of *user dimensions*, e.g. $\Delta = \{\text{experience, skill, goal\_orientation, presentation\_preferences, training}\}$. For each dimension $\delta \in \Delta$ we assume to be given a scale $sc(\delta)$. Formally, a *scale* is a totally ordered set.

**Example 3.** The scale for goal-orientation may be

$$sc(\text{goal-orientation}) = \{\text{surfer, navigator, searcher}\}$$

with surfer $\leq$ navigator $\leq$ searcher. As another example consider

$$sc(\text{presentation\_preferences}) = \{\text{detailed, normal, condensed, terse}\}$$

with detailed $\leq$ normal $\leq$ condensed $\leq$ terse.

If $\Delta = \{\delta_1, \ldots, \delta_n\}$ is a set of user dimensions, the *set of user profiles* over $\Delta$ is $gr(\Delta) = sc(\delta_1) \times \cdots \times sc(\delta_n)$. A *user type* over $\Delta$ is a convex region $U \subseteq gr(\Delta)$.

## 5 Modelling Tasks

The primary purpose of a WIS is to provide information to its users. This information is usually used in order to perform a certain task. Such tasks can be performed by a single

user or they can be the cooperative effort of several users. That is, we consider WIS to be task-oriented systems.

A task in a home loan system can be to submit a mortgage application. This involves only one actor in a single role customer. In an extended system, the task may be to submit, approve and implement a mortgage, in which case other actors in the role of a bank clerk or a mortgage advisor would participate in the task.

Tasks describe the general purposes of the WIS. They combine roles that are involved in the task, actions executed by actors in these roles, consequently scenes to which these actions belong, information consumed by the actions, and data flowing between the actions. In addition, there is an event that triggers the task. Actions can be grouped together into subtasks to provide a more concise form of task specification. Thus, a task is largely specified by what has already been defined for the story space including the integrity constraints that define rights, obligations, intentions and behaviour.

Formally, a *task* $\tau$ consists of a set $act(\tau) = \{\tau_1, \ldots, \tau_n\}$ of subtasks, which may be atomic actions in the story space and a triggering event $ev(\tau)$, which is the combination of a boolean condition $\varphi$ on the story space and the fact that a particular action $\alpha$ was executed by some role $r$, i.e. $ev(\tau) = (\varphi, do(r, \alpha))$.

Furthermore, we will associate with each subtask $\tau_i$ a set of scenes and a set of roles. If $\tau_i$ is atomic, i.e. an action $\alpha$, then is will be associated with exactly one scene $s$ and exactly one role $r$.

**Example 4.** Look again at the loan application system from [SKWM03]. In this system we may have a task application_for_mortgage with triggering event

$$ev(\tau) = (\text{TRUE}, do(\text{customer}, \text{enter\_loan\_system})),$$

i.e. the task is triggered when a customer enters the loan system.

Then the set of subtasks contains activities such as select_mortgage, describe_object, select_m_terms_and_conditions, etc.

## 6 Personalisation of the Story Space

The problem of story space personalisation according to the preferences of a particular WIS user exploits the preference rules that have been defined for a user profile. For instance, if there is a (conditional or unconditional) preference for a particular action in case of a choice, the less-preferred option can be discarded from the story space.

Personalisation also means satifying the intention of a particular WIS user. This can be formalised by using the *goal*, i.e. a postcondition to the story space. Then the general dependencies can be exploited to simplify the story space.

**Example 5.** In Example 1 we may have to deal with a user who already knows everything about loans. That means that condition $\varphi_5$ is true throughout the navigation through the

story space. Furthermore, $\varphi_5$ excludes conditions $\varphi_0$, $\varphi_1$ and $\varphi_2$. Using this, the story space can be simplified to

enter_loan_system ;
( select_home_loan $\{\varphi_6\}$ $\Box$ select_mortgage $\{\varphi_7\}$ ) ;
( ( $\{\varphi_6\}$ ( provide_applicant_details ;
( provide_applicant_details $\Box$ `skip` ) ;
( describe_loan_purpose $\|$ enter_amount_requested $\|$
enter_income_details ) ;
select_hl_terms_and_conditions ) $\{\varphi_8\}$ ) $\Box$
( $\{\varphi_7\}$ ( provide_applicant_details ; provide_applicant_details$^*$ ;
( describe_object $\|$ enter_mortgage_amount $\|$
describe_securities$^*$ ) ;
( enter_income_details $\|$ enter_obligations$^*$ ) ;
( ( $\{\neg\varphi_{12}\}$ select_m_terms_and_conditions ;
calculate_payments )$^*$ ;
$\{\varphi_{12}\}$ select_m_terms_and_conditions ) ) $\{\varphi_9\}$ ) ) ;
confirm_application $\{\varphi_{10} \vee \varphi_{11}\}$

Similarly, we may have to look at a user who intends to apply for a home loan. This can be expressed by the goal $\varphi_{10}$. In this case $\varphi_{10}$ excludes $\varphi_{11}$ and $\varphi_9$. This implies that the story space can be simplified to

enter_loan_system ;
( ( $\{\varphi_0\}$ look_at_loans_at_a_glance $\Box$
( $\{\varphi_1\}$ request_home_loan_details ;
( look_at_home_loan_samples $\Box$ `skip` ) $\{\varphi_3\}$ ) $\Box$
( $\{\varphi_2\}$ request_mortgage_details ;
( look_at_mortgage_samples $\Box$ `skip` ) $\{\varphi_4\}$ ) )$^*$ $\{\varphi_5\}$ ) ;
select_home_loan $\{\varphi_6\}$ ;
( ( provide_applicant_details ;
( provide_applicant_details $\Box$ `skip` ) ;
( describe_loan_purpose $\|$ enter_amount_requested $\|$
enter_income_details ) ;
select_hl_terms_and_conditions ) $\{\varphi_8\}$ )
confirm_application $\{\varphi_{10}\}$

By using also the fact that the goal $\varphi_{10}$ excludes $\varphi_2$ the story space can be further reduced to

enter_loan_system ;
( ( $\{\varphi_0\}$ look_at_loans_at_a_glance $\Box$
( $\{\varphi_1\}$ request_home_loan_details ;
( look_at_home_loan_samples $\Box$ `skip` ) $\{\varphi_3\}$ ))$^*$ $\{\varphi_5\}$ ) ;
select_home_loan $\{\varphi_6\}$ ;

( ( provide_applicant_details ;
( provide_applicant_details □ skip ) ;
( describe_loan_purpose || enter_amount_requested ||
enter_income_details ) ;
select_hl_terms_and_conditions ) $\{\varphi_8\}$ )
confirm_application $\{\varphi_{10}\}$

The work in [ST04] exploits equational reasoning with Kleene algebras with tests for the purpose of WIS personalisation.

## 7 Conclusion

In this article we presented a central part of a conceptual modelling approach to web information systems dealing with storyboarding, which addresses the following problems:

- modelling tasks that users will perform either individually or cooperatively while using the WIS;
- modelling the paths a potential user may take through the WIS including the actions performed along such a path;
- modelling obligations and rights of users in a particular role;
- modelling user profiles and preference rules that arise from them;
- modelling the intention of users.

Furthermore, we showed how to personalise a WIS to preferences and intentions of users.

The most challenging future research direction to be continued is to approach extensions to the propositional reasoning about the storyboard and to widen the scope towards an inclusion of deontic constraints that are used to model obligations and rights. Furthermore, we think of widening the reasoning scope also by switching from propositional reasoning to general dynamic logic. This means to take also the updates of the underlying databases into consideration.

## References


[AGS98] Atzeni, P., Gupta, A., and 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.

[BCFM00] Bonifati, A., Ceri, S., Fraternali, P., and Maurino, A.: Building multi-device, content-centric applications using WebML and the W3I3 tool suite. In: *ER Workshops 2000*. volume 1921 of *LNCS*. pp. 64–75. Springer-Verlag. Berlin. 2000.

[BGP00] Baresi, L., Garzotto, F., and 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.

[BKST04] Binemann-Zdanowicz, A., Kaschek, R., Schewe, K.-D., and Thalheim, B.: Context-aware web information systems. In: Hartmann, S. and Roddick, J. (Eds.), *Conceptual Modelling 2004 – First Asia-Pacific Conference on Conceptual Modelling*. volume 31 of *CRPIT*. pp. 37–48. Dunedin, New Zealand. 2004. Australian Computer Society.

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

[CFP99] Ceri, S., Fraternali, P., and 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.

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

[FKST00] Feyer, T., Kao, O., Schewe, K.-D., and Thalheim, B.: Design of data-intensive web-based information services. In: Li, Q., Ozsuyoglu, Z. M., Wagner, R., Kambayashi, Y., and Zhang, Y. (Eds.), *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE 2000)*. pp. 462–467. IEEE Computer Society. 2000.

[FP98] Fraternali, P. and 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.

[Fr99] Fraternali, P.: Tools and approaches for developing data-intensive web applications: A survey. *ACM Computing Surveys*. 31(3):227–263. 1999.

[FT99] Feyer, T. and Thalheim, B.: E/R based scenario modeling for rapid prototyping of web information services. In: Chen, P. P.-S. (Ed.), *Advances in Conceptual Modeling*. volume 1727 of *LNCS*. pp. 253–263. Springer-Verlag. 1999.

[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., and Matthews, C.: Story boarding for web information systems. In: Taniar, D. and Rahayu, W. (Eds.), *Web Information Systems*. pp. 1–33. IDEA Group. 2004.

[MMA99] Mecca, G., Merialdo, P., and Atzeni, P.: ARANEUS in the era of XML. *IEEE Data Engineering Bullettin*. 1999.

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

[RSL99] Rossi, G., Schwabe, D., and 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.

[SKWM03] Schewe, K.-D., Kaschek, R., Wallace, C., and Matthews, C.: Modelling web-based banking systems: Story boarding and user profiling. In: *Advanced Conceptual Modeling Techniques: ER 2002 Workshops*. volume 2784 of *LNCS*. pp. 427–439. Springer-Verlag. 2003.

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

[ST01] Schewe, K.-D. and Thalheim, B.: Modeling interaction and media objects. In: Bouzeghoub, M., Kedad, Z., and Métais, E. (Eds.), *Natural Language Processing and Information Systems: 5th International Conference on Applications of Natural Language to Information Systems, NLDB 2000*. volume 1959 of *LNCS*. pp. 313–324. Springer-Verlag. Berlin. 2001.

[ST04] Schewe, K.-D. and Thalheim, B.: Reasoning about web information systems using story algebras. submitted for publication. 2004.