Web Information Systems as Abstract State Services

WISM'08

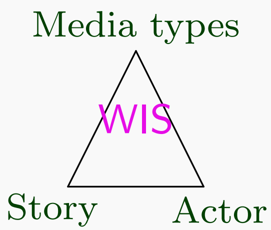
Barcelona

22. 10. 2008

Hui Ma, Klaus-Dieter Schewe, Bernhard Thalheim, Qing Wang

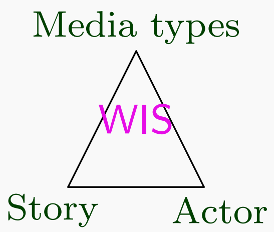
Victoria, Kiel Universities, New Zealand & Germany

Information Science Research Centre,Palmerston North, New Zealand



Overview

- Why — Challenges, deficiencies, opportunities, hopes
- WIS ?!? — Web **information systems**
  web services for information intensive systems
- Foundations — Semantic foundations, reasoning, validation, verification, proofs, evolution
- States — Appropriate depending on the user, component
- Services — Towards a modern theory of web services
- Personalisation — One of the main challenges nowadays



Background and Motivation

Specification framework for flexible web services

Theoretical foundation of web services

Reasoning framework

Mapping facilities for infrastructure, database, network etc. support

Challenges

Contextualisation as the biggest task nowadays

Personalisation as the ultimate support for any system

Specification for a flexible, configurable (CMDB), high-quality web system

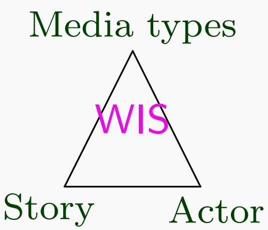
Integration on actual demand and context

Foundation and Validation, Verification, Proof

What we did achieve so far?

Storyboarding for web information systems

Media types as the main mediating structures between a user and a web information system



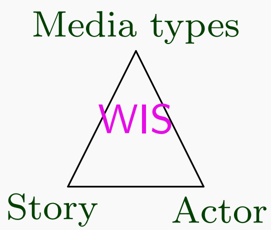
Our Goals

Starting Points

- Take an abstract, conceptual approach to service integration, composition and personalisation

- A (data-intensive) service can be described by two layers:
  - a hidden database layer consisting of a database schema and transactions
  - a visible view layer on top of it providing views and functions based on them

- This idea appears for dialogue systems, Web Information Systems, component-based systems



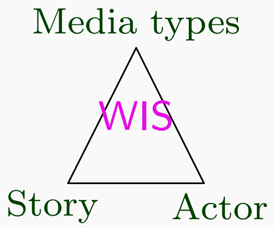
The Idea of Abstract State Services

- Define a concept of abstract services for modern web information systems

- Develop a theory of Abstract State Services (ASSs) following the line of thought of the ASM thesis (Blass, Gurevich)
  - Formalise sequential and parallel algorithms by a small set of intuitive, abstract postulates
  - Prove that these postulates are always satisfiable by (sequential) ASMs

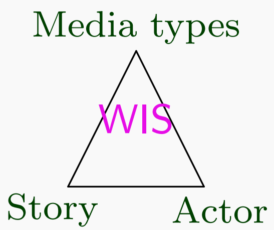
- Carry this idea over to database transformations (queries, updates)

- This forms the basis of the formal definition of ASSs by means of postulates



Service Integration, Composition, Personalisation

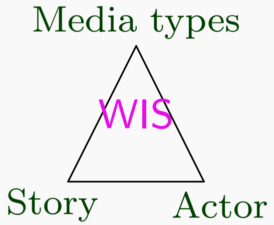
- Integration means to replace two or more ASSs by a single new one that offers all the functionality of the individual services
- Reduced integration to database schema and view integration
- Composition requires the extraction of service components from existing ASSs that feed a new service without replacing the original ones
- We also need composition operations
- Personalisation directs the extraction of suitable components by preferences



Web Information Systems

From Simple Web to Web Information Systems

- Web Information System (WIS) is a database-backed information system that is realised and distributed over the web with user access via web browsers
- A WIS is open to (almost) every user, but access in a particular role may be restricted
- Websites have gone through three stages: Web 1.0, Web 2.0, Web 3.0
  - Web1.0 is mainly author-driven
  - Web2.0 is user driven and content centered
  - Web3.0 is characterised by (4C + P + VS)



Web 1.0

Our past, present and also future: Web 1.0 has mainly been oriented towards content provision to deliver content with a rudimentary functionality, e.g., navigation, acquisition information, linking, search and browse

Achievements of Web 1.0 e.g.

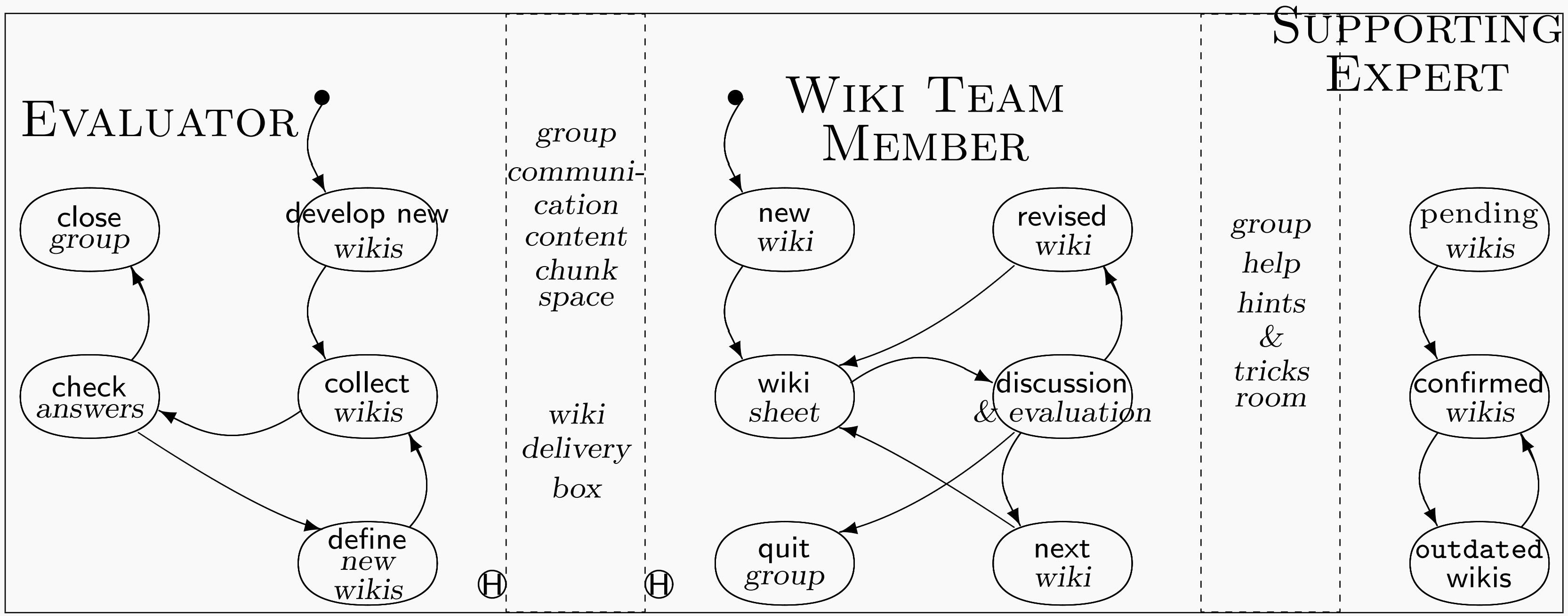
- Resulting websites are simple to use, without any learning effort
- Based on an application development according to application scenarios
- It models story spaces as schemes for utilization
- It uses security techniques provided by server-sided logic and aiming being robust
- It uses taxonomies for classification and systematisation

Limitations of Web 1.0 e.g.

- It is often resulted in high time exposure for stepwise page buildup
- It is based on fixed page content
- It uses information dissemination from a central source
- It pushes information to users
- It is based on a unidirectional communication
- It does not providing relation to desktop organization

Example 1: Wiki Storyboards

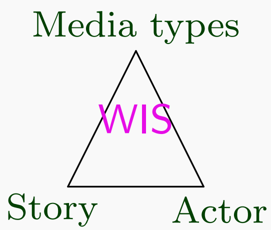
What we did achieve already!!



forming a Wiki team with three roles (evaluator, member, supporter)

modelling the Wiki collaboration story

modelling the intended result



Example 2: Actors in a Paper Submission and Reviewing System

PC (Co-)Chair: via chair login

- list of overviews, all data, abstracts and papers
- paper assignment, deblock of data, PC decisions

PC Member: via personal member login with abstract/paper download

- before assignments: abstracts of submitted papers
- entire own review
- before PC session, after completing reviews: concurrent reviews
    - conflicts with concurrent reviews, discrepancies
- during PC session: anonymous survey reviews
- before assignments: indicate levels of interest
- until PC session: input / modify reviews for assigned papers
- until PC session: obtain / submit review template for assigned papers

Author: via entry page or personal author login

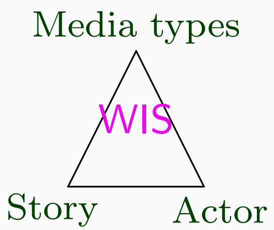
- general and his data
- before submission deadline: submit abstract / paper
- after PC session: obtain anonymous reviews, submit final version

Administrator: via admin tools and direct access (encrypted data)

- maintenance of software code, database, password update

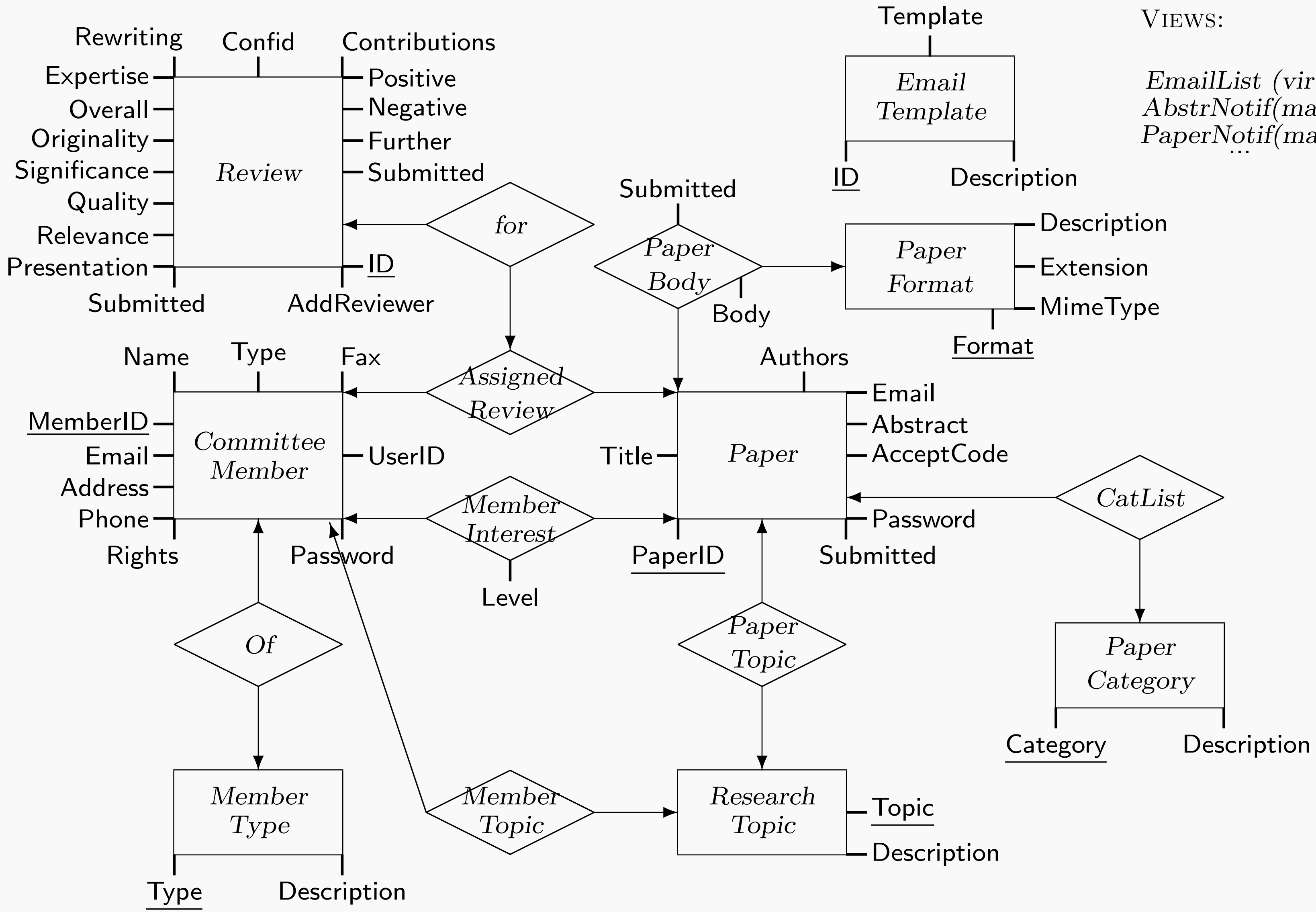
Abstract
State
Services
22. 10. 2008
B. Thalheim

Overview
Why
WIS ?!?
Foundations
DB Postulate
Services
Personalisation
Finally



Example 2: Data Structuring for a Paper Submission and Reviewing System

PCPhases: CMEditProfile; CMBrowseAbstr; ReviewsDue; CMViewOther; Session
AuPhases: ASubmAbstr; ANotifAbstr; ASubmPaper; ANotifPap; Decision; FinalPDue
PCOrgPhases: AbstractCheck; PaperCheck; ReviewsContr; Assignment; #Reviews



Views:

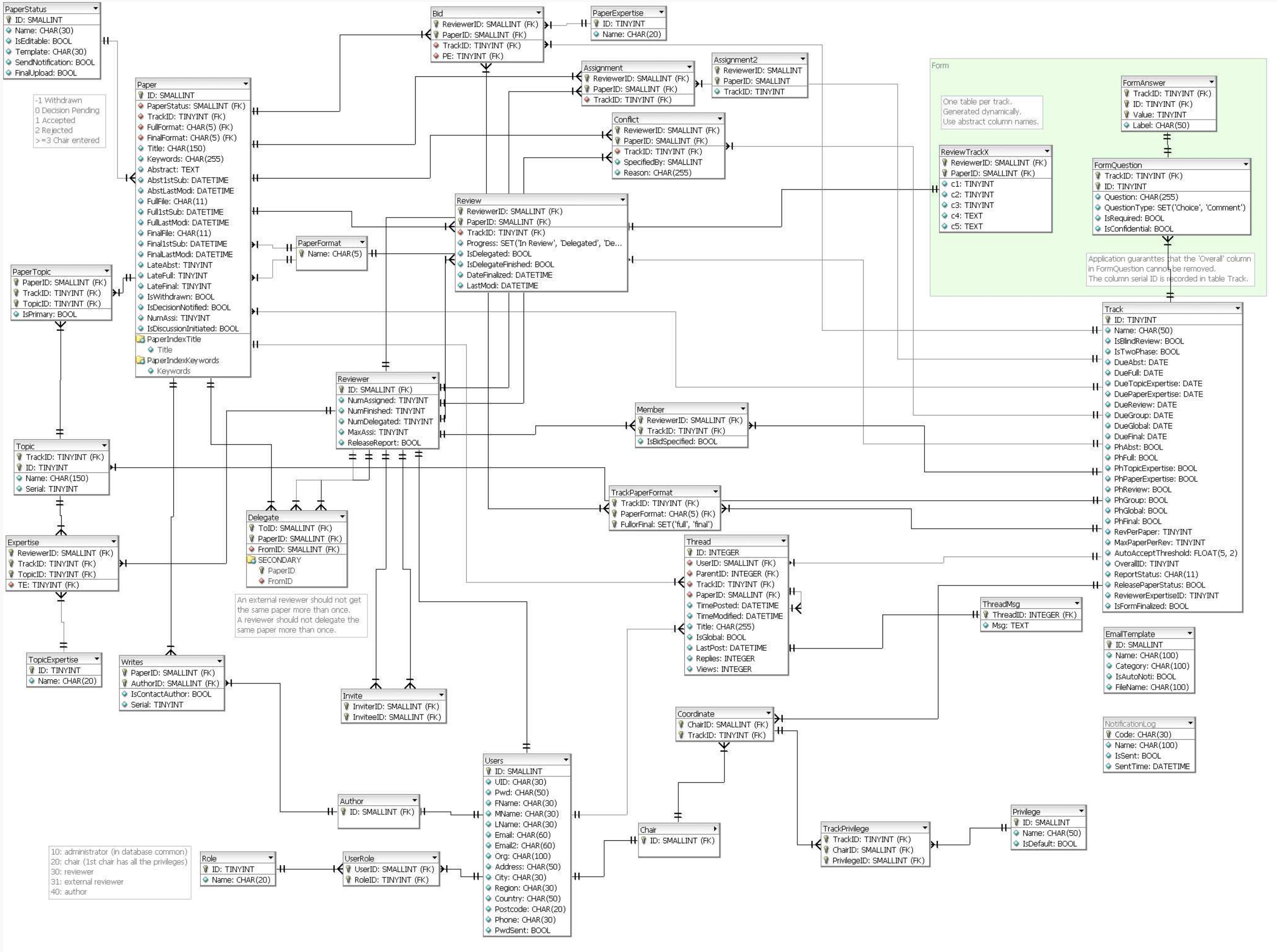
EmailList (virt)
AbstrNotif(mat)
PaperNotif(mat)
...

Example 2: Another Data Structure

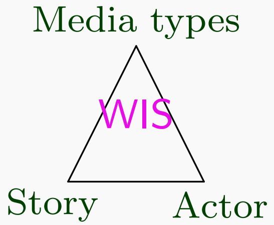
Abstract State Services
22. 10. 2008
B. Thalheim

Overview
Why
WIS ?!?
Foundations
DB Postulate
Services
Personalisation
Finally

Media types
WIS
Story
Actor



PaperReviewSubmissionMuComsDatabaseSchema.eps



Example 2:

Submission and Reviewing System: Media Type *(PCMemberEntry)*

parameterized expression on

InformationMediaType, EnabledManipulationRequest, SuppliedProcesses

as triple for each component of the media object

composition via

bounded iteration, sequential, branching, conditions, ...

SequentialExpression *with Parameters (userID,pwrd)*

object(Conference head)

PC Member Work Sheet *link to download information*

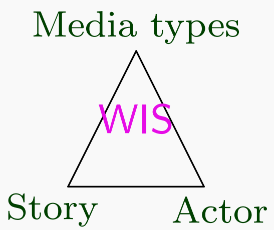
(`media type` *AssignedPapers*, $\emptyset$, $\emptyset$)
(`media type` *Missing Reviews*, $\emptyset$, $\emptyset$)
(`media type` *ReviseSubmitted*, *ReplaceByRevision*, {*Select*, *RevisionLink*})
(`media type` *ViewSubmitted*, $\emptyset$, $\emptyset$)
`if` *CommitteeMember* $\in$ *chair* `then` (`media type` *ReviewByPC*, $\emptyset$, *LinkToRbPY*)
if CommitteeMember $\in$ *chair* `then` (`media type` *ReviewStatus*, $\emptyset$, *LinkToRS*)
if AllReviewsCompleted `then` (`media type` *AnomymousReview*, $\emptyset$, *LinkToAR*)
`if` *CommitteeMember* $\in$ *chair* `then` (`media type` *AssignReviewers*, $\emptyset$, *LinkToAssRev*)
`if` *CommitteeMember* $\in$ *chair* `then` (`media type` *ViewAllPapers*, $\emptyset$, *DownloadPaper*)
`if` *time* $\in$ *BrowseOpen* `then` (`media type` *BrowseSubmittedAbstracts*, $\emptyset$, $\emptyset$)
`if` *time* $\in$ *BrowseOpen* `then`(`media type` *BrowseTitlesOnly*, $\emptyset$, $\emptyset$)

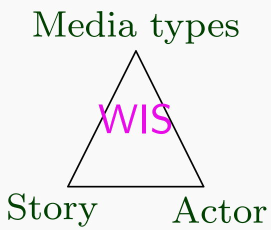
Example 2:
Paper Submission and Reviewing System: Services

`Media Type` (View, Functionality (, View Interfaces))

`MediaType` *ConferenceInformation(ConfXYZ,LinkSet)*
`MediaType` *AssignedPapers(*
    AssignedReview(,CommitteeMember(MemberID)),*
    DownloadPaper)
`MediaType` *MissingReviews(*`if` *MissingReviews(memberID)*
    `then` *for(...)* `else` *Thanks,* $\emptyset$*)*
`MediaType` *ReviseSubmitted(SubmittedReviews = for(AssignedReview(),Review),* $\emptyset$*)*
`MediaType` *ViewSubmitted(SubmittedReviews = for(AssignedReview(),Review),* $\emptyset$*)*
`MediaType` *ReviewsByPC()*
`MediaType` *ReviewStatus()*
`MediaType` *AnonymousReview()*
`MediaType` *AdHocSQL()*
`MediaType` *AssignReviewers()*
`MediaType` *ViewAllPapers*
`MediaType` *BrowseSubmittedAbstracts(*$\emptyset$*,getAbstractList* $\setminus$ *forbidden)*
`MediaType` *BrowseTitlesOnly(*$\emptyset$*,getAbstractList* $\setminus$ *forbidden)*
`MediaType` *ConferenceInformation(ConfXYZ,LinkSet)*

Overview
Why
WIS ?!?
Foundations
DB Postulate
Services
Personalisation
Finally





Example 2: Paper Submission and Reviewing System: Media Object *(submitForm)*

Structure *(pID,pwrd)* — COMPILED VERSION

object(Conference head)

Submission form for papers — *link to help information*
information on visibility problems

First part - Submission of your paper — *(mandatory)*

Format selection for your paper - select one of possible choices

Location of your paper — *- browse* $\vee$ *type*

Second part - check/correct the data you submitted with abstract — *(optional)*

Contact person — f_1 *(first name, last name, phone, fax, email, complete surface address)*

Title of paper = f(Paper(Title))

Author_list (#, first name, last name, name of company or institute)

PC Member (is any of the authors member of the PC) (yes/no)

Remarks — *(optional)*

Submit data for step 2 — *reset* — *(either)*

Functions

observe_correction(Part#2)

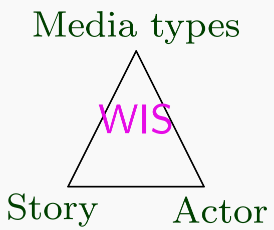
transfer("address/file")

collect(Address,Name,Format,Extension)

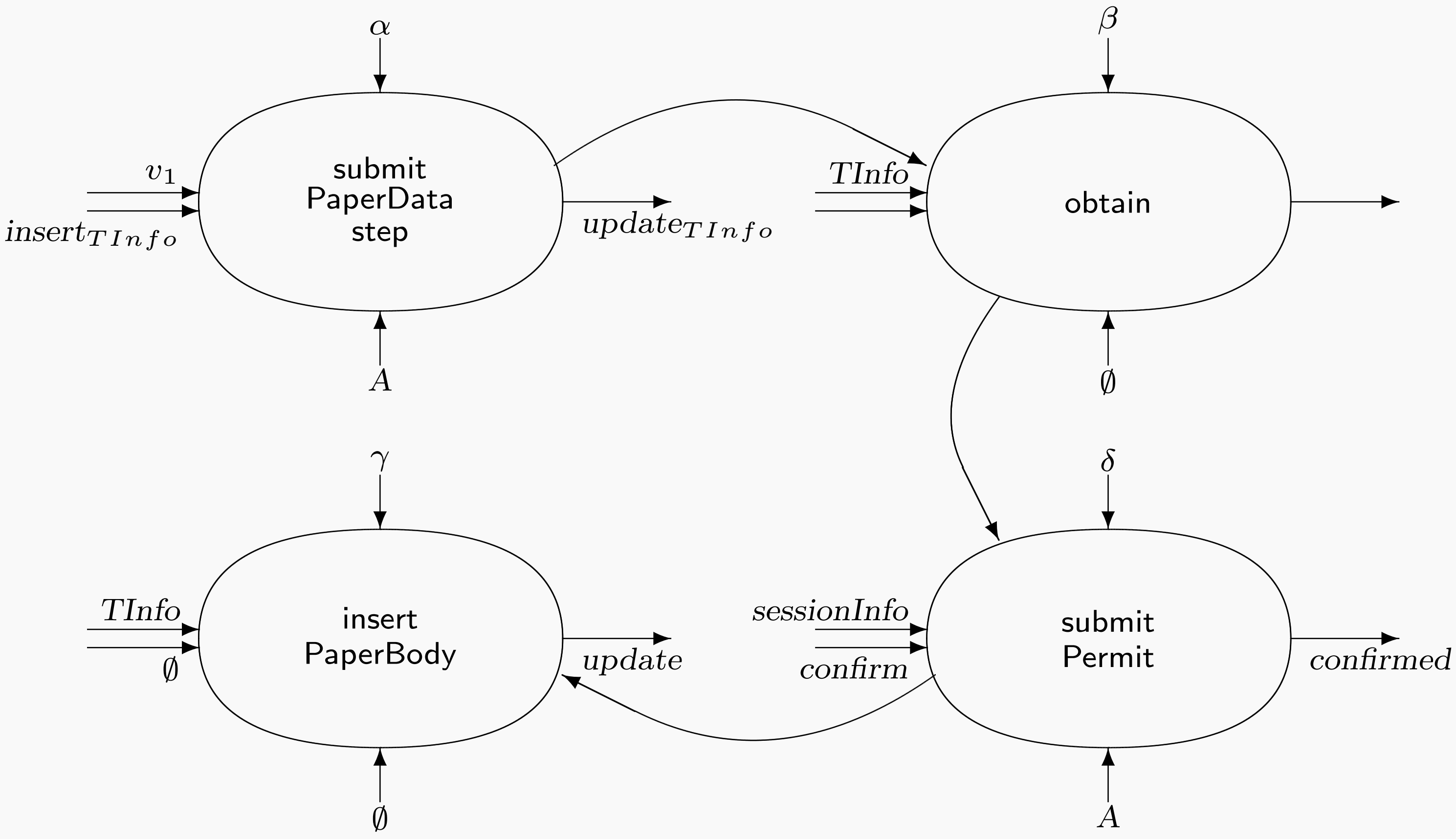
$insert_{PaperBody}(pID, format, Body, today)$

$observe\_correct(Part\#2) \quad \longrightarrow \quad update_{Paper}(pID, Authors, Title, Email)$

Abstract State Services
22. 10. 2008
B. Thalheim



Example 2: Paper Submission and Reviewing System: Dialogue Scene



$v_1 = \sigma_{PaperID=pID}$(PaperBody)

α = on LinkPaperSubmit if deadline = ok $\wedge$ Login = ok
accept on submitConfirmed $\wedge$ $\neg$ collectError

β = on submitPaper if paperBody = ok accept on extension = ok

γ = accept on $\neg$ transferError

δ = on submitPaper if 1 accept on confirmed

A = actor[pID,pwrd]

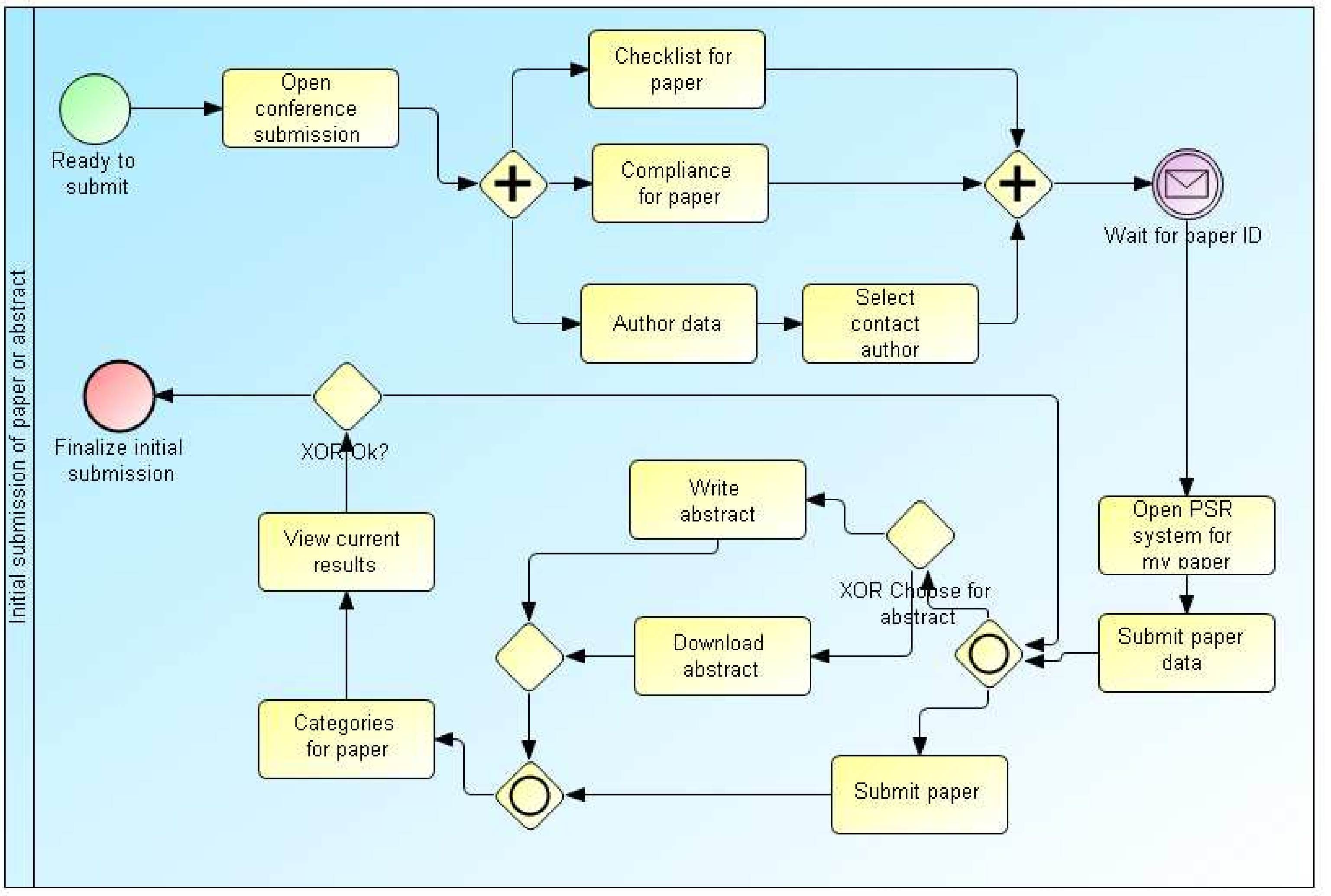
Example 2: Mapping to Workflow-Specification, e.g. BPMN 1.0

Abstract State Services
22. 10. 2008
B. Thalheim

Overview
Why
WIS ?!?
Foundations
DB Postulate
Services
Personalisation
Finally

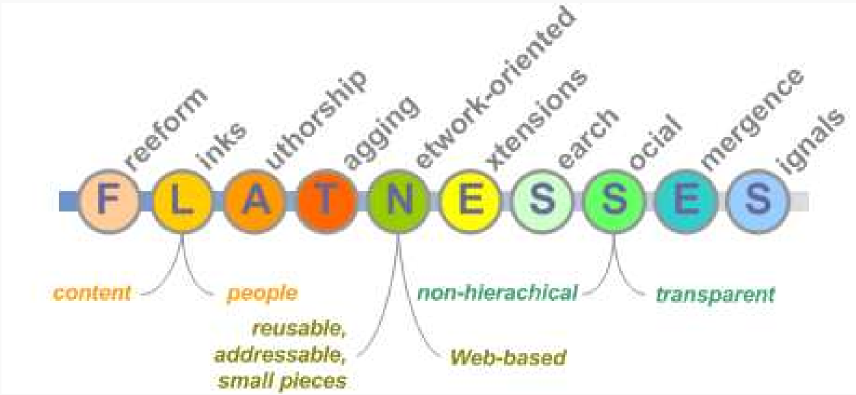
Media types
WIS
Story
Actor

Example Initial Paper Submission for Paper submission and reviewing system

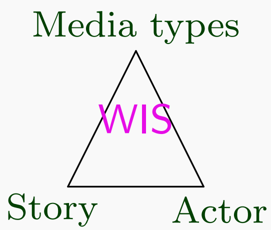


Services Requested by Web Technologies

Generic Functions, Platforms, Services ... towards Enterprise 2.0

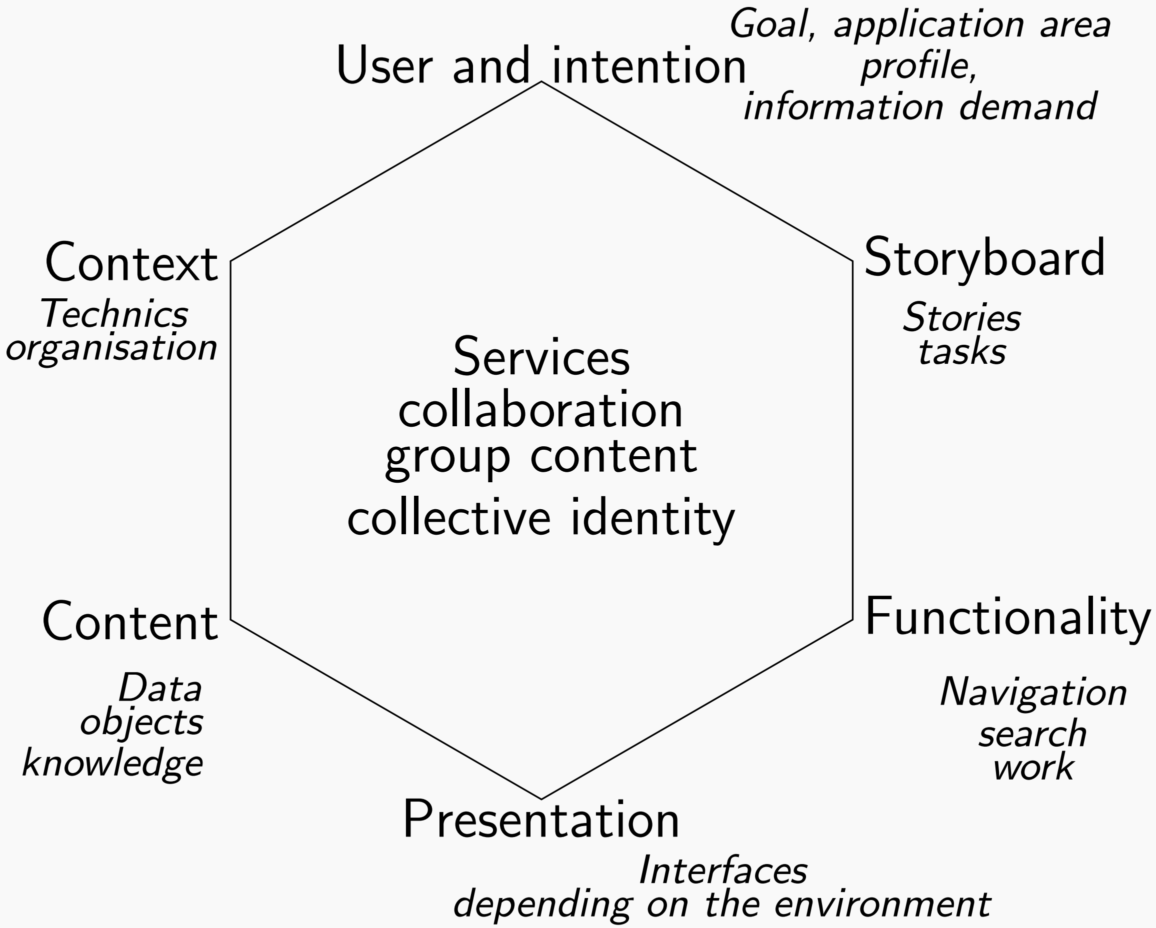


Media types
WIS
Story Actor

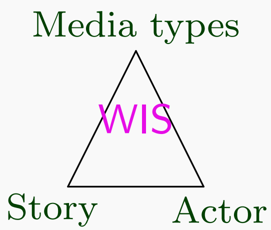


The Co-Design Hexagonal Dimensions

Towards sophisticated web engineering

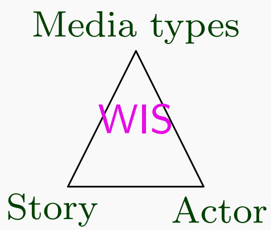


Services allow context injection and is user-centered and story-centered



Main Challenge of Web 1.0, 2.0, 3.0, ...: Development of Services on top of DBS

- Database architecture distinguishes at least three layers:
  - a conceptual layer describing the database schema in an abstract way
  - a physical layer implementing the schema – not relevant for us here
  - an external layer made out of views
- For data-intensive services complete this architecture by adding operations:
  - On the conceptual layer operations are handled as database transactions
  - On the external layer the operations provide the means with which users can interact with a database



Media Types, Media Object Suite

Theoretical Basis

- Interaction types $= (ct(M), q_M, Op_M)$
  content type $cont(M)$, defining query q_M
  generic functions Op_M for changing the database
- Attached operations: (signature, selection type, body)
  selection type - supertype of $ct(M)$
  e.g. generalization/specialization, reordering, browsing, linking, surveying, searching, join
- Media type: interaction type $+$ unit extension
  $+$ order extension $+$ cohesion/adhesion $+$ hierarchical versions
- Usage modelling: usage dimensions, scales, user profiles, user kind, context, session
- Container = (ct(C), layout(C), kind(C))
  for shipping and representation

Database Systems and their Specification

Database system state $\mathcal{DBS}$

(input states $\mathcal{I}$, output states $\mathcal{O}$, DBMS states $\mathcal{E}$, database states $\mathcal{D}$)

Modification programs :

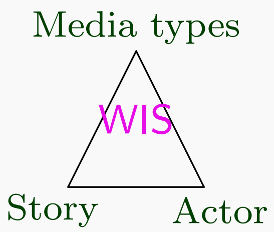
if $\mathcal{I}(req) \neq \lambda \wedge \mathcal{E}(modify) = enabled \wedge \mathcal{I}(req) \in Update$
then $\mathcal{I}(req) := \lambda$, $\mathcal{O}(errMsg) := \ldots$, $\mathcal{D} := \ldots$, $\mathcal{E} :=$
...

Retrieval rules :

if $\mathcal{I}(req) \neq \lambda \wedge \mathcal{E}(retrieve) = enabled \wedge \mathcal{I}(req) \in Update$
then $\mathcal{I}(req) := \lambda$, $\mathcal{O}(errMsg) := \ldots$, $\mathcal{O}(answer) := \ldots$

DBMS control rules :

if $\mathcal{E}(DBMSstateChange)$ and $\mathcal{E}(modify) = disabled$
then $\mathcal{D} := \ldots$, $\mathcal{E} := \ldots$



Database Systems and their Specification

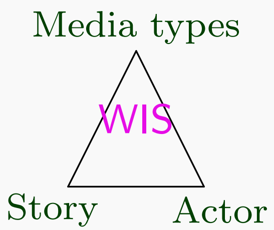
Database system state $\mathcal{DBS}$

(input states $\mathcal{I}$, output states $\mathcal{O}$, DBMS states $\mathcal{E}$, database states $\mathcal{D}$)

DBS operation rules

- ModifyInput(request,DBMS_state, DB_state),
- ModifyOutput(request,DBMS_state, DB_state),
- ModifyDB(request,DBMS_state, DB_state),
- ModifyControl(request,DBMS_state, DB_state),
- RetrieveOutput(request,DBMS_state, DB_state),
- RetrieveDB(request,DBMS_state, DB_state),
- RetrieveControl(request,DBMS_state, DB_state),
- ControllerDBMS(DBMS_state, DB_state), and
- ControllerDB(DBMS_state, DB_state).

`modify :`	$(req,\_,s,d)$	$\mapsto$	$(\_,errMsg,s',d')$
`retrieve :`	$(req,\_,s,d)$	$\mapsto$	$(\_,answ \cup errMsg,s,d)$
`controller :`	$(\_,\_,s,d)$	$\mapsto$	$(\_,\_,s',d')$



Layers of Database Systems Specification

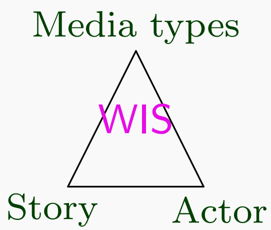
Level 1 (Point of view of business users)**:** Database system defined by three state space: input state, database state, output state. The *input state* is based on algebraic structure with ground terms defined on the values and the names. The *database state* is based on the (object-)relational structure with well-defined composition operators. The *output state* is a general database defined on the values and names.

Level 2 (Conceptual point of view)**:** Database systems are defined as an extension of level 1 by transactions, constraints, views and integrity maintenance.

Level 3-1 (Logical point of view)**:** The logical database system defined as an extension of level 2 by states of the database management system by the transaction and recovery engine, by the synchronization engine, the logging engine and the query translating engine.

Level 3-2 (Physical point of view)**:** The physical database system is defined as an extension of level 3-1 by specific functions of the DBMS.

Level 4 (DBMS point of view)**:** On level 4, the storage engine is modelled in detail with the buffers and the access engine.



Database Postulate

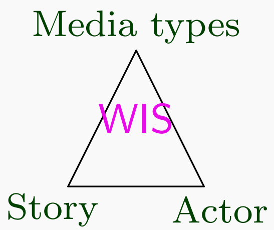
A **database system** DBS consists of

- a set $\mathcal{S}$ of states, together with a subset $\mathcal{I} \subseteq \mathcal{S}$ of initial states,
- a wide-step transition relation $\tau \subseteq \mathcal{S} \times \mathcal{S}$, and
- a set $\mathcal{T}$ of transactions, each of which is associated with a small-step transition relation $\tau_t \subseteq \mathcal{S} \times \mathcal{S}$ $(t \in \mathcal{T})$ satisfying the postulates of a database transformation over $\mathcal{S}$.

A **run** of a database system DBS is an infinite sequence $S_0, S_1, \dots$ of states $S_i \in \mathcal{S}$:

- starting with an initial state $S_0 \in \mathcal{I}$
- for all $i \in \mathbb{N}$ $(S_i, S_{i+1}) \in \tau$ holds
- for all $i \in \mathbb{N}$ there is a transaction $t_i \in \mathcal{T}$ with a finite run $S_i = S_i^0, \dots, S_i^k = S_{i+1}$ such that $(S_i^j, S_i^{j+1}) \in \tau_{t_i}$ holds for all $j = 0, \dots, k-1$.

Overview
Why
WIS ?!?
Foundations
DB Postulate
Services
Personalisation
Finally



Abstract State Services (ASS) Based on the Extended View Postulate

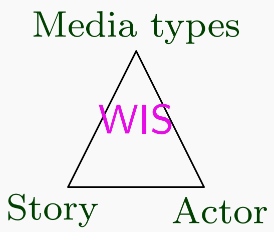
An **Abstract State Service** (ASS) consists of

- a database system DBS, in which each state $S \in \mathcal{S}$ is a finite composition $S_d \cup V_1 \cup \cdots \cup V_k$, and
- a finite set $\mathcal{V}$ of (extended) views.

Each view $v \in \mathcal{V}$ is associated with a database transformation such that for each state $S \in \mathcal{S}$ there are views $v_1, \ldots, v_k \in \mathcal{V}$ with finite runs $S_d = S_0^j, \ldots, S_{n_j}^j = S_d \cup V_j$ of v_j $(j = 1, \ldots, k)$.

Each view $v \in \mathcal{V}$ is further associated with a finite set $\mathcal{O}_v$ of (service) operations $o_1, \ldots, o_n$ such that for each $i \in \{1, \ldots, n\}$ and each $S \in \mathcal{S}$ there is a unique state $S' \in \mathcal{S}$ with $(S, S') \in \tau$.

Furthermore, if $S = S_d \cup V_1 \cup \cdots \cup V_k$ with V_i defined by v_i and o is an operation associated with v_k, then $S' = S'_d \cup V'_1 \cup \cdots \cup V'_m$ with $m \geq k - 1$, and V'_i for $1 \leq i \leq k - 1$ is still defined by v_i.



Foundations in a Nutshell

Postulates for database transformations: sequential time, abstract state, background, exploration boundary, genericity

First-order structures for state description and semantics

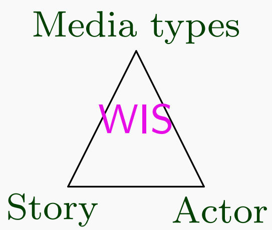
Abstract state postulate for abstraction from all machine, coding, ... assumptions

Background of a computation based on background classes

Location operators for abstraction from storage

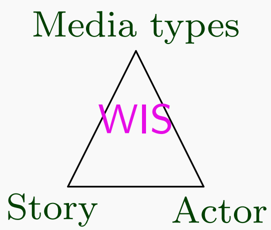
Update sets

Overview
Why
WIS ?!?
Foundations
DB Postulate
Services
Personalisation
Finally



Service Integration

- The integration of ASSs aims at replacing two given ASSs by a new one that supports the functionality of both original ASSs
- Start with an integration of database systems DBS^1 and DBS^2:
  - Use another set $\mathcal{S}\!\downarrow$ of states together with projection functions $p : \mathcal{S}^1 \to \mathcal{S}\!\downarrow$ and $q : \mathcal{S}^2 \to \mathcal{S}\!\downarrow$.
  - p and q should be invariant under isomorphisms, i.e. isomorphic states are to be mapped to isomorphic states
  - state integration requires the existence of a set $\mathcal{S}\!\uparrow$ of integrated states together with projection functions $\bar{p} : \mathcal{S}\!\uparrow \to \mathcal{S}^1$ and $\bar{q} : \mathcal{S}\!\uparrow \to \mathcal{S}^2$ that are "universal" in the following sense:
    - The diagram defined by p, q, $\bar{p}$ and $\bar{q}$ permutes, i.e. $p \circ \bar{p} = q \circ \bar{q}$ holds
    - For any other set of states $\mathcal{S}$ together with projections $p' : \mathcal{S} \to \mathcal{S}^1$ and $q' : \mathcal{S} \to \mathcal{S}^2$ that satisfy $p \circ p' = q \circ q'$ there exists a unique function $r : \mathcal{S} \to \mathcal{S}\!\uparrow$ with $\bar{p} \circ r = p'$ and $\bar{q} \circ r = q'$



Effects on Transactions and Views

- This "pullback" definition is that it carries over to the transactions and the transition relations:
  - If S_1 is the start state of a transaction $t \in \mathcal{T}^1$, then we have a transition $(S_1, S_1') \in \tau^1$ that is defined by a run of t
  - Let $S \in \mathcal{S}\!\uparrow$ be a state that results from integrating S_1 with $S_2 \in \mathcal{S}^2$, then the corresponding function $\bar{p}$ maps S onto S_1
  - There is a state $S' \in \mathcal{S}\!\uparrow$ that results from integrating S_1' with S_2
  - The pair (S, S') is the natural extension of (S_1, S_1') to a transition on states $\mathcal{S}\!\uparrow$
- The definition also allows us to preserve views:
  - Let v be a view on $\mathcal{S}^1$, which transforms a state S_d^1 into a state $S_d^1 \cup V^1$
  - If S_d is a state after integrating S_d^1 with some state S_d^2 originating from DBS^2, then $S_d \cup V^1$ results from integrating $S_d^1 \cup V^1$ with $S_d^2 \cup V^1$

Integrating Views

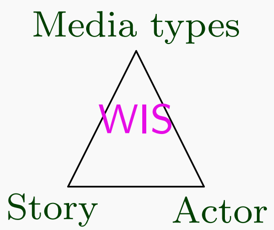
- Not only interested in preserving views, but in integrating them, which can be approached in the same way:
  - For views v^1 on DBS^1 transforming S^1 into $S^1 \cup V^1$ and v^2 on DBS^2 transforming S^2 into $S^2 \cup V^2$ we can first integrate S^1 and S^2 into the integrated state S
  - This turns v^1 and v^2 both into views over S
  - We can then separately integrate V^1 and V^2 into V, which means that we can replace v^1 and v^2 by an integrated view that will transform S into $S \cup V$
- Finally, operations associated with a view carry over to the views after integration, as they merely induce a transaction and a change to the active views

Service Composition

- Composition of ASSs does not aim at replacing any existing ASS
- The goal is to define new services that exploit functionality of existing ones
- We have to extract components from existing ASSs and recompose these components

Let $\mathcal{A} = (DBS, \mathcal{V}) = (\mathcal{S}, \tau, \{\tau_t\}_{t \in \mathcal{T}}, \{(v, \{o_1, \dots, o_{n_v}\})\}_{v \in \mathcal{V}})$ be an ASS.

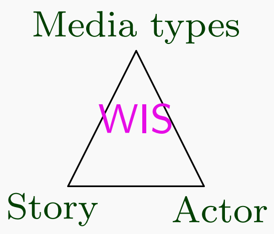
A **component** of $\mathcal{A}$ is an ASS $(\mathcal{S}, \tau, \{\tau_t\}_{t \in \mathcal{T}}, \{(p_v \circ v, \{o'_1, \dots, o'_{n'_v}\})\}_{v \in \mathcal{V}'})$ with $\mathcal{V}' \subseteq \mathcal{V}$ and $\{o'_1, \dots, o'_{n'_v}\} \subseteq \{o_1, \dots, o_{n_v}\}$.



Parallel Composition of Components

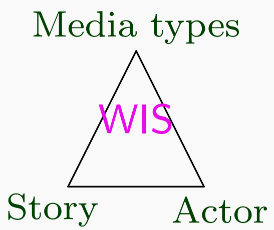
Let $\mathcal{A}^i = (\mathcal{S}^i, \tau^i, \{\tau_t\}_{t \in \mathcal{T}^i}, \{(v, \{o_1, \dots, o_{n_v^i}\})\}_{v \in \mathcal{V}^i})$ $(i = 1, \dots, n)$ be ASSs. Their **parallel composition** $\mathcal{A}^1 \oplus \cdots \oplus \mathcal{A}^n$ is an ASS that is defined as follows:

- The set of states is the sum $\mathcal{S} = \{S_1 \cup \cdots \cup S_n \mid S_i \in \mathcal{S}^i\}$.
- The wide-step transition relation τ is defined by parallel composition, i.e. $(S_1 \cup \cdots \cup S_n, S_1' \cup \cdots \cup S_n') \in \tau$ iff $(S_i, S_i') \in \tau^i$ for all $i = 1, \dots, n$.
- The set of transactions is the product $\mathcal{T} = \{t_1 \| \dots \| t_n \mid t_i \in \mathcal{T}^i\}$.
- Small step transition relations are defined by parallel composition, i.e. $(S_1 \cup \cdots \cup S_n, S_1' \cup \cdots \cup S_n') \in \tau_{t_1 \| \dots \| t_n}$ iff $(S_i, S_i') \in \tau_{t_i}$ for all $i = 1, \dots, n$.
- The set of views is also defined as a product $\mathcal{V} = \{v_1 \| \dots \| v_n \mid v_i \in \mathcal{V}^i\}$.
- The sets of service operations are defined by parallel composition $\mathcal{O}_{v_1 \| \dots \| v_n} = \{o_1 \| \dots \| o_n \mid o_i \in \mathcal{O}_{v_i}\}$.



Service Personalisation

- How can the selection process (in component extraction) be tailored so that out of the views and associated operations on offer only those are selected that are relevant for the intended use
- Concentrate on the service operations treating the views they are associated with as necessary basis
- To support the automatic or semi-automatic selection of service operations from a given ASS we have to know how it is supposed to be used
- For this purpose we associate an action scheme or plot with an ASS
- A plot will be an algebraic expression composed out of the service operations together with Boolean pre- and postconditions that prescribes meaningful sequences of operations



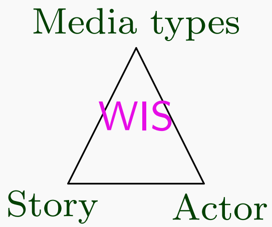
Plots

Let $\mathcal{O}$ denote the set of service operations associated with an ASS $\mathcal{A}$, and let $\mathcal{C}$ be a set of Boolean conditions.
The set $\mathcal{P}$ of **plots** over $\mathcal{O}$ and $\mathcal{C}$ is the smallest set with $\mathcal{O} \cup \mathcal{C} \cup \{0, 1\} \subseteq \mathcal{P}$ satisfying the following conditions:

- For $p, q \in \mathcal{P}$ we also have $pq \in \mathcal{P}$, $p+q \in \mathcal{P}$, $p\|q \in \mathcal{P}$ and $p^* \in \mathcal{P}$.
- For $p \in \mathcal{P}$ not involving any operation in $\mathcal{O}$ we also have $\bar{p} \in \mathcal{P}$.

Assumptions for plots:

- A Boolean condition is combined with an operation that tests it
- $\mathcal{P}$ must satisfy the axioms of Kleene algebras with tests

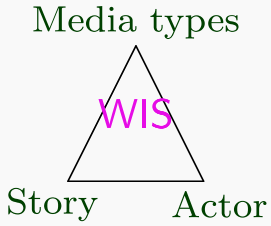


Preference Rules

Define preference rules by means of equations on $\mathcal{P}$, e.g.:

- $\alpha(p+q) = \alpha p$ means that under the condition α, if there is a choice between p and q, then p will be preferred.
- $p(q+r) = pq$ means that after p, if there is a choice between q and r, then q will be preferred.
- $\alpha p^* = \alpha p$ means that under the condition α the preference is to execute p exactly once instead of iterating it arbitrarily often.
- $\bar{\alpha} p = 0$ means that α is a precondition for p.
- $p \bar{\alpha} = 0$ means that α is a postcondition for p.

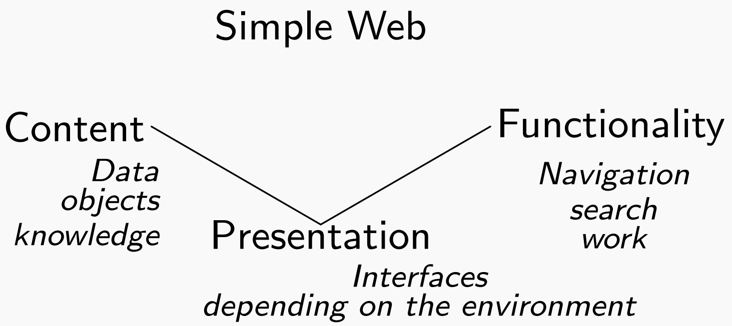
Together with the conditional equations that define the axioms for Kleene algebras with tests we can use the given plot p and a postcondition β that we want to reach (it could simply be 1), and apply the equations as term rewriting rules to turn $p\beta$ into a simpler form, say p'.

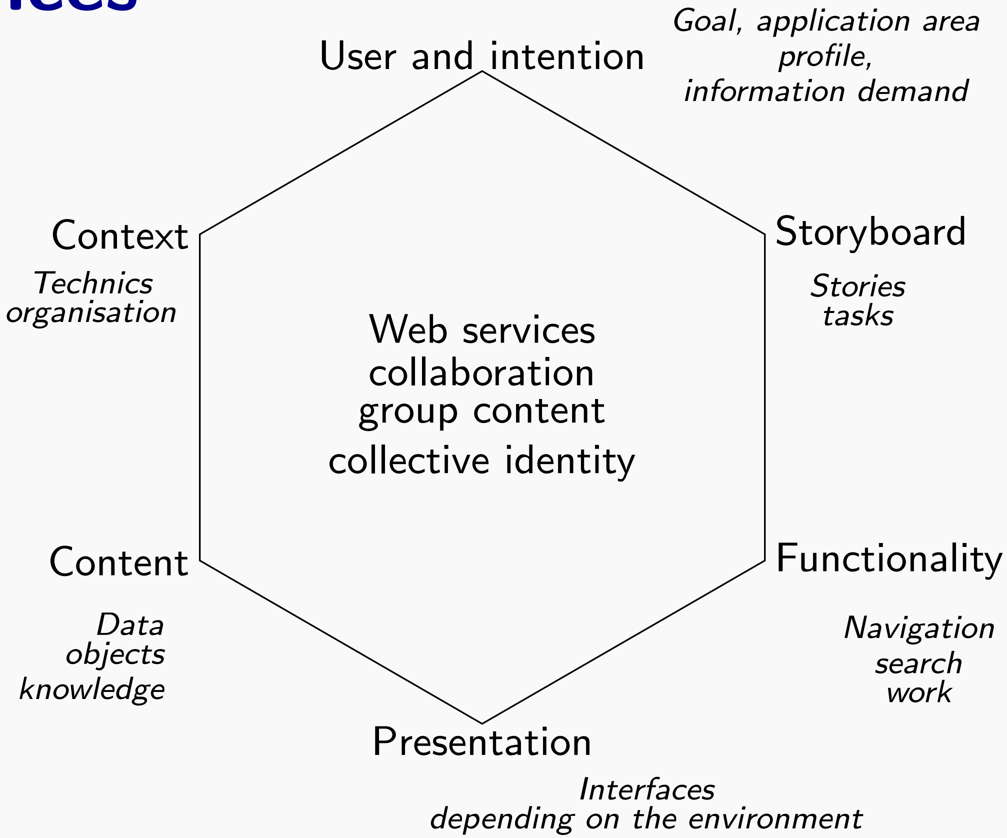


Concluding

- Languages: Define languages that satisfy the postulates / capture ASSs
- Service Composition: Further explore ways to compose ASS components
- Personalisation: Lift the rewriting approach from a propositional level to one that uses conditions on views
- Applications: Elaborate in detail how ASSs can be used for web warehousing, composition of WISs and other web services, etc.
- "Applied" Applications: Elaborate case studies for ASS use in web interoperability

From Simple Web Applications to Web Services



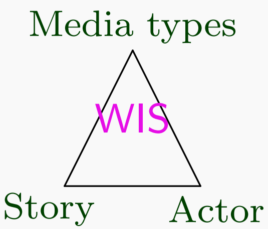


Web 1.0: author driven,
publish/provide_story/support or
advertise/wait/attract/react/retain
for users:
inform/subscribe/obtain/answer/come_back

Web 2.0: user driven, content centered, GoogleAdSense, Flickr, **Wikipedia**, blogs, optimised search engines, pay per click, web services, participate instead be attracted, tagging, syndication
common usage of bookmarks, clicks; communities; tracking goals of usage; data ownership, portability, economics, transparency; architectures of participation
Aal principle: *andere arbeiten lassen*

Thank you!

thalheim@is.informatik.uni-kiel.de



ASM in a Nutshell (1): States

A **signature** Σ is a finite collection of function names.

- Each function name f has an arity, a non-negative integer.
- Nullary function names are called constants.
- Function names can be static or dynamic.
- Every ASM signature contains the static constants *undef, true, false*.

A **state** $\mathcal{A}$ for the signature Σ is a non-empty set X, the superuniverse of $\mathcal{A}$, together with an interpretation $f^{\mathcal{A}}$ of each function name f in Σ.

- If f is an n-ary function name of Σ, then $f^{\mathcal{A}} : X^n \to X$.
- If c is a constant of Σ, then $c^{\mathcal{A}} \in X$.
- The superuniverse X of the state $\mathcal{A}$ is denoted by $|\mathcal{A}|$.

Relations are functions that have the value *true, false, undef*

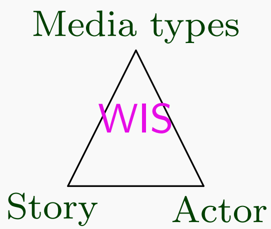
$$(a \in R \text{ iff } R(a) = true).$$

$dom(f^{\mathcal{A}}) = \{(a_1, ..., a_n) \in |\mathcal{A}| \mid f^{\mathcal{A}}(a_1, ..., a_n) \neq undef\}$

The superuniverse can be divided into subuniverses

represented by unary relations.

Overview
Why
WIS ?!?
Foundations
DB Postulate
Services
Personalisation
Finally



ASM in a Nutshell (2): Updates

A **location** $\mathcal{A}$ is a pair $\boxed{l = (f, (a_1, ..., a_n))}$

- $f^{\mathcal{A}}(a_1, ..., a_n)$ - content of the location in $\mathcal{A}$
- $\{a_1, ..., a_n\}$ - set of elements of the location $\mathcal{A}(l)$

An **update** for $\mathcal{A}$: $u = (l, v)$

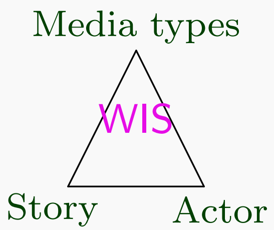
- trivial if $v = \mathcal{A}(l)$
- update set: set of updates

An **update** set U is consistent, if it has no clashing updates, i.e., if for any location l and all elements v, w if $(l, v), (l, w) \in U$ then $v = w$

The **result of firing** a consistent update set U: new state $\mathcal{A} + U$

$$(\mathcal{A} + U)(l) = \begin{cases} v & \text{if } (l, v) \in U \\ \mathcal{A}(l) & \text{if there is no } v \text{ with } (l, v) \in U \end{cases}$$

for all l of $\mathcal{A}$

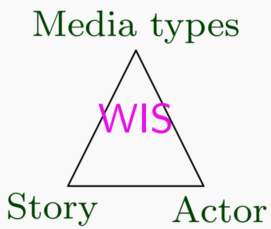


ASM in a Nutshell (3): Models and Transitions

A state $\mathcal{A}$ is a **model** of ϕ ($\mathcal{A} \models \phi$) if $[\![\phi]\!]^{\mathcal{A}}_{\zeta} = \mathit{true}$ for all variable assignments ζ for ϕ.

A transition rules Σ

- Skip rule: $\boxed{\textbf{skip}}$
- Update rule: $\boxed{f(s_1, ..., s_n) := t}$
- Parallel execution rule: $\boxed{P \textbf{ par } Q}$
- Conditional rule: $\boxed{\textbf{if } \phi \textbf{ then } P \textbf{ else } Q}$
- Let rule: $\boxed{\textbf{let } x = t \textbf{ in } P}$
- For all rule: $\boxed{\textbf{forall } x \textbf{ with } \phi \textbf{ do } P}$
- Choose rule: $\boxed{\textbf{choose } x \textbf{ with } \phi \textbf{ do } P}$
- Sequence rule: $\boxed{P \textbf{ seq } Q}$
- Call rule: $\boxed{r(t_1, ..., t_n)}$



ASM in a Nutshell (4): Abstract State Machine

A **rule declaration** for a rule name r of arity n is an expression $\boxed{r(x_1, ..., x_n) = P}$ where

- P is a transition rule and
- the free variables of P are contained in the list $x_1, ..., x_n$.

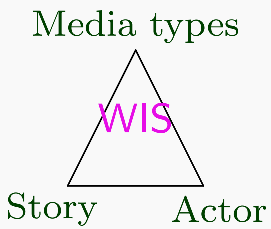
An **abstract state machines** M consists of a

- a signature Σ,
- a set of initial states for Σ,
- a set of rule declarations,
- a distinguished rule name of arity zero called *main rule name* of the machine.

The transition rule P yields the update set U in a state $\mathcal{A}$ under the variable assignment ζ: `yields`(P,$\mathcal{A}$,ζ,U).

Semantics of transition rules defined in a calculus by rules:

$$\frac{\text{Premise}_1, ..., \text{Premise}_n}{\text{Conclusion}}\ \text{Condition}$$



Further Reading /1

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

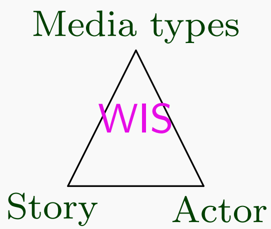
(2) Binemann-Zdanowicz, A., Schulz-Brünken, B., Thalheim, B., and Tschiedel, B. Flexible e-payment based on content and profile in the e-learning system DaMiT. In *Proceedings of E-Learn 2003* (Phoenix (Arizona), USA, 2003), Association for the Advancement of Computing in Education.

(3) Binemann-Zdanowicz, A., Thalheim, B., and Tschiedel, B. Logistics for learning objects. In *Proceedings of eTrain 2003* (2003).

(4) Düsterhöft, A. and Thalheim, B. Linguistic based search facilities in snowflake-like database schemes. Data Knowl. Eng. 48(2): 177-198 (2004)

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

(6) Feyer, T., Schewe, K.-D., and Thalheim, B. Conceptual modelling and development of information services. In *Conceptual Modeling – ER'98*, T. Ling and S. Ram, Eds., vol. 1507 of *LNCS*. Springer-Verlag, Berlin, 1998, pp. 7–20.



Further Reading /2

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

(8) Fiedler, G., and Thalheim, B. An Approach to Conceptual Schema Evolution Preprint 0701, Department of Computer Science, Kiel University, Januar 2007

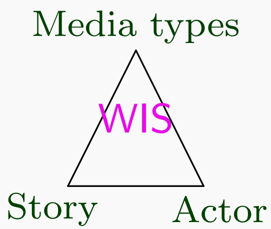
(9) Hui Ma, Schewe, K.-D., and Thalheim, B. Integration and Cooperation of Media Types. ISTA 2005: 139-153

(10) Hui Ma, Schewe, K.-D., and Thalheim, B. Context Analysis: Toward Pragmatics of Web Information Systems Design. APCCM 2008, 69-78.

(11) Hui Ma, Schewe, K.-D., and Qing Wang Distribution design for higher-order data models. Data Knowl. Eng. 60(2): 400-434 (2007).

(12) Hui Ma, Schewe, K.-D., Thalheim, B. and Zhao, J. View Integration and Cooperation in Databases, Data Warehouses and Web Information Systems. J. Data Semantics IV: 213-249 (2005)

(13) Kaschek, R., Matthews, C., Schewe, K.-D., and Wallace, C. Analyzing web information systems with the Abstraction Layer Model and SiteLang. In *Proceedings of the Australasian Conference on Information Systems (ACIS 2003)* (2003).



Further Reading /3

(14) Kaschek, R., Schewe, K.-D., Thalheim, B., and Zhang, L. Integrating context in conceptual modeling for web information systems. In *Web Services, E-Business, and the Semantic Web, Revised Selected Papers of Second International Workshop WES 2003* (2004), C. Bussler, D. Fensel, M. E. Orlowska, and J. Yang, Eds., vol. 3095 of *LNCS*, Springer-Verlag, pp. 77–88.

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

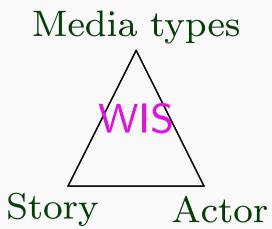
(16) Moritz, T., Schewe, K.D., and Thalheim B. Strategic Modelling of Web Information Systems and Its Impact on Visual Design Patterns. International Journal on Web Information Systems, vol. 1 (4), pp. 77-94, 2005.

(17) Noack, R. and Thalheim, B. Patterns for screenography. LNBIB5, Springer, 2008, 484-495

(18) Riaz-ud-Din, F., Noack, R., Schewe, K.D., Hui Ma, and B. Thalheim Capturing forms in information systems design. In *Innovations'07*, IEEE 2007

(19) Schewe, K.-D. Querying web information systems. In *Conceptual Modeling – ER 2001*, H. S. Kunii, S. Jajodia, and A. Sølvberg, Eds., vol. 2224 of *LNCS*. Springer-Verlag, 2001, pp. 571–584.

(20) Schewe, K.-D. The power of media types. In *Web Information Systems - WISE 2004*, X. Zhou, S. Su, M. Papazoglou, M. Orlowska, and K. Jeffery, Eds., vol. 3306 of *LNCS*. Springer, 2004, pp. 53–58.



Further Reading /4

(21) Schewe, K.-D. and Thalheim, B. An Algorithmic Approach to High-Level Personalisation of Web Information Systems. Proc. Web Age Information Management (WAIM 2005), LNCS vol. 3739, pp. 737-742.

(22) 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* (2003), vol. 2784 of *LNCS*, Springer-Verlag, pp. 427–439.

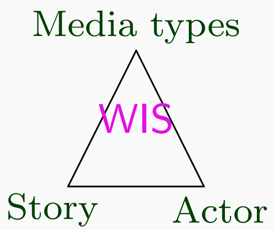
(23) Schewe, K.-D., Kaschek, R., Wallace, C., and Matthews, C. Emphasizing the Communication Aspects for the Successful Development of Electronic Business Systems. *Information Systems and E-Business Management*, vol. 3(1): 71-100, 2005.

(24) Schewe, K.-D., and Schewe, B. Integrating database and dialogue design. *Knowledge and Information Systems 2*, 1 (2000), 1–32.

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

(26) Schewe K.D. and Thalheim B. The Co-Design Approach to Web Information Systems Development. International Journal on Web Information Systems, vol. 1(1): pp. 5-14, 2005.

(27) Schewe K.D. and Thalheim B. Usage-Based Storyboarding for Web Information Systems Preprint 0613, Department of Computer Science, Kiel University, November 2006



Further Reading /5

(28) Schewe K.D. and Thalheim B. Life Cases: A Kernel Element for Web Information Systems Engineering. WEBIST (Selected Papers) 2007: 139-156

(29) Schewe, K.-D., and Thalheim, B. User Models: A Contribution to Pragmatics of Web Information Systems Design. WISE 2006: 512-523

(30) Schewe, K.-D., and Thalheim, B. Reasoning about web information systems using story algebras. *Proceedings ADBIS 2004* (Budapest, Hungary, 2004), LNCS 3255, 54-66, 2004.

(31) Schewe, K.-D., and Thalheim, B. Structural media types in the development of data-intensive web information systems. In *Web Information Systems*, D. Taniar and W. Rahayu, Eds. IDEA Group, 2004, pp. 34–70.

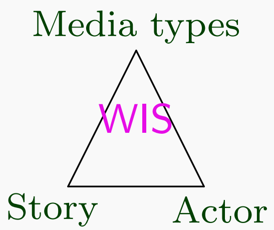
(32) Schewe, K.-D., and Thalheim, B. Term Rewriting for Web Information Systems - Termination and Church-Rosser Property. WISE 2007: 261-272

(33) Schewe, K.-D., and Thalheim, B. Personalisation of web information systems - A term rewriting approach. Data Knowl. Eng. 62(1): 101-117 (2007)

(34) Schewe, K.-D., and Thalheim, B. Conceptual modelling of web information systems. *Data and Knowledge Engineering*, 54 (2005), 147-188

(35) Schewe, K.-D., Zhao, J., and Thalheim, B. Quality Assurance in Web Information Systems Development. QSIC 2007: 219-224

(36) Schewe, K.-D., Thalheim, B., and Zlatkin, S. Modelling Actors and Stories in Web Information Systems. ISTA 2004: 13-23



Further Reading /6

(37) Schewe, K.-D., and Thalheim, B. Pragmatics of storyboarding for web information systems: Usage analysis International Journal Web and Grid Services, 128-169 (2007)

(38) Schewe, K.-D., Thalheim, B., Binemann-Zdanowicz, A., Kaschek, R., Kuss, T., and Tschiedel, B. A conceptual view of electronic learning systems. *Education and Information Technologies*, vol. 10 (1-2): pp. 83-110, 2005

(39) Schewe, K.-D., and Thalheim, B. Facets of Media Types. LNBIB5, Springer, 2008, 296-305

(40) Thalheim, B., and Düsterhöft, A. The use of metaphorical structures for internet sites. *Data & Knowledge Engineering 35* (2000), 161–180.

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

(42) Thalheim, B. *Entity-relationship modeling – Foundations of database technology*. Springer, Berlin, 2000. See also http://www.is.informatik.uni-kiel.de/∼thalheim/HERM.htm.

(43) Thalheim, B. Component development and construction for database design. Data and Knowledge Engineering, July 2005, Vol. 54, Issue 1, 77-95

(44) Thalheim, B., Al-Fedaghi, S.S., Al-Saqabi, K. Information Stream Based Model for Organizing Security. ARES 2008: 1405-1412