Florian Matthes, Christian Neubert, Alexander W. Schneider

Fostering Collaborative and Integrated Enterprise Architecture Modelling

Enterprise Architecture Management (EAM) is a challenging task in modern enterprises. Tools supporting EA management activities are based on extensive models either created by enterprise architects or built-in. However, these models cannot be adapted according to specific data acquisition needs by persons having the architectural knowledge on the instance level. In this article, we describe how Hybrid Wikis empower these information carriers and enterprise architects to collaboratively and incrementally develop and manage a model in a bottom-up fashion by using wiki pages enriched with types and attributes. We visualise these emergent models by using UML-like class diagrams. Our approach is evaluated in a case study in a global operating bank participating in our Wiki4EAM community, a community of German enterprise architects applying Hybrid Wikis in different EA management contexts.

1 Motivation & Problem Statement

The field of Enterprise Architecture (EA) management constantly evolves since the often cited publication of Zachman (1987). Until today, the application of different subsequently published approaches did not result in the achievement of all promised benefits in practice. For example, Lucke et al. (2010) report that communication, shared understanding, and insufficient tool support are still critical issues in EA management. Tool support is often provided by means of heavyweight and expensive EA management tools to gather, structure, visualise, and analyse architectural information, such as business processes, applications, and organisational units. In order to obtain a holistic and consistent view of the EA to be managed it is required to define the concepts existing within an enterprise formally (cf. Bunge 1977). Although foundational ontologies exist for EAM frameworks currently applied in practice, for example ArchiMate (Azevedo et al. 2011) and TOGAF (Gerber et al. 2010), their meta-models need to be tailored (Källgren et al. 2009) to reflect the specific organisational context. In this situation often the so-called ivory tower syndrome (Raadt et al. 2008) emerges. That is, enterprise architects ‘invent’ an information model intended to be filled with data by the employees of the technical departments (Buckl et al. 2009b). However, these models are often rather unsuitable for the data acquisition as required by the persons having the concrete architectural knowledge as they are too abstract or too complex. Due to the high amount of data necessary to provide a holistic EA description many organisations today use specialised EAM tools (Matthes et al. 2008). The actual data input is thereby often performed by many different employees, because the architectural knowledge about single instances is spread over different employees within the company.

Our approach evaluates ways of bringing information carriers having deeper insights in an enterprise’s structure within their individual scope and enterprise architects closer together by providing an integrated environment to collaboratively develop and manage both models and instances. It facilitates instance and model co-evolution by using lightweight techniques, such as suggestions. Information carriers are enabled to create and adapt the description of instances according to their specific needs. From these instances a model can be derived in a second step. Additionally, enterprise architects can explicitly define a model. For example, an architect can introduce new elements not yet represented by instances (e.g., introduce a new attribute) or introduce elements frequently used in instances but not yet defined in the enterprise model. Both models (i.e., derived and defined) are presented in a combined view helping enterprise architects to reflect their design decisions regarding the model. This way our approach mitigates the ivory tower syndrome.

The remainder of this article is structured as follows. In Sect. 2 we present a bottom-up approach for the creation of EA management models which uses Hybrid Wikis as concept and tool (Matthes et al. 2011). In order to enable enterprise architects to manage the resulting emergent and collaboratively created model, we introduce a suitable extension to UML class diagrams in Sect. 3. In Sect. 4 we present the findings of a case study with a globally operating bank, in order to demonstrate the feasibility of our approach for bottom-up and collaborative EA model creation. We demarcate our approach from other approaches for EA model creation in Sect. 5. Finally, Sect. 6 summarises this article and Sect. 7 provides a critical reflection and outlook on future research.

2 Approach

To address the challenges introduced in Sect. 1 Matthes et al. (2011) developed the concept of Hybrid Wikis and implemented it. Hybrid Wikis combine traditional wikis with a small set of structuring concepts in order to allow querying like in an object-oriented database. The provided structuring concepts are: attributes, types, attribute definitions, and constraints. These concepts are maintained (created, deleted, changed) by wiki users.

2.1 Structured Wiki Pages

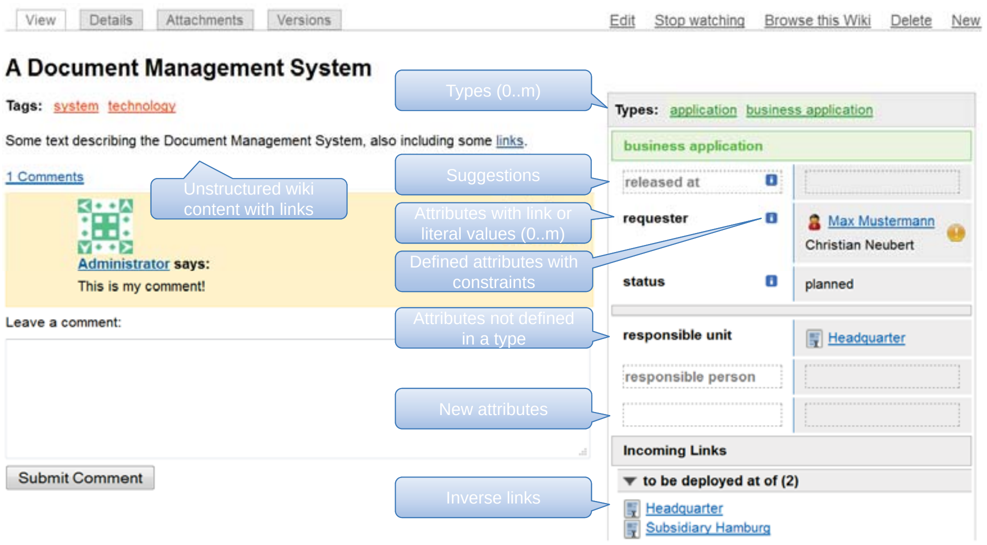
Attributes and type assignments enrich individual wiki pages with structured information. For instance, a page can be typed with *business application* and provide an attribute *status* with value *planned* (cf. Fig. 1). A typed page represents an instance of a type. The data type of an attribute value can either be a literal (Text, Date, Number) or a hyperlink to another (typed) page. An attribute can be multivalued (i.e., an ordered list of values). Attributes can be freely added and removed to and from pages independent of the currently assigned types. The same applies to types, that is, a type, can be removed from a page without changing the currently assigned attributes. This means that types and attributes are basically independent from each other.

Wiki pages structured with attributes and type assignments represent the instances.

2.2 Underlying Model and its Validation

Types, attribute definitions, and constraints are elements representing the model of all instances. By means of attribute definitions attribute keys (e.g., *status* in Fig. 1) can be bound to a type. Constraints belong to an attribute definition and allow the specification of value ranges for attributes. A constraint, for example, can require that the values of an attribute (e.g., *responsible unit*) are links to pages having a specific type (e.g., *organisational unit*). A page's attributes (and their values) are validated by means of a constraint if the constraint's attribute definition is related to the respective attribute. That is, if their keys are matching (e.g., both have key *responsible unit*) and the page uses the same type the attribute definition belongs to. Constraint violations are indicated to the user on wiki page level by showing a validation message (cf. *requester* Fig. 1). However, users are never forced to enter valid values when changing an invalid page. Therefore constraints are referred to as 'soft' in Hybrid Wikis.

Additionally, types and attribute definitions can be declared as strict. Thus, changes to a page can only be stored if all attributes of this page are valid. However, in Hybrid Wikis everybody allowed to modify wiki pages (with attributes and type assignments) can also change types,


View
Details
Attachments
Versions
Edit
Stop watching
Browse this Wiki
Delete
New
Types (0..m)
Suggestions
Unstructured wiki content with links
Attributes with link or literal values (0..m)
Defined attributes with constraints
Attributes not defined in a type
New attributes
Inverse links
Submit Comment

A Document Management System

Tags: system technology

Some text describing the Document Management System, also including some links.

1 Comments

Administrator says:
This is my comment!

Leave a comment:

Types: application business application

business application

released at	
requester	Max Mustermann Christian Neubert
status	planned
responsible unit	Headquarter
responsible person	

Incoming Links

to be deployed at of (2)
Headquarter
Subsidiary Hamburg

Figure 1: *A wiki page provided by Hybrid Wikis*

attribute definitions, and constraints. Therefore, when using strict types (or attribute definitions) users are rather urged to enter valid values only, but never forced.

In Hybrid Wikis instances and their model can diverge, even when using strict constraints. This is due to the fact that constraints can be defined even if violating pages result from this definition. But due to the loose coupling of both they can be changed independently without being restricted by each other. This way instance and model evolution is facilitated.

In Matthes et al. (2011) and Neubert (2012) all modelling elements of Hybrid Wikis, as shown in Fig. 2, are explained in detail. Additionally, it is explained how these concepts are implemented based on an existing wiki system in terms of, for example, algorithms, performance, system architecture, and technology. An important fact is, that attributes are related directly to wiki pages without the indirection of a type. Nevertheless, a type can be used to group attributes via the usage of attribute definitions. By this way of attribute modelling, users are enabled to assign them without thinking about types while modellers can still use familiar modelling concepts.

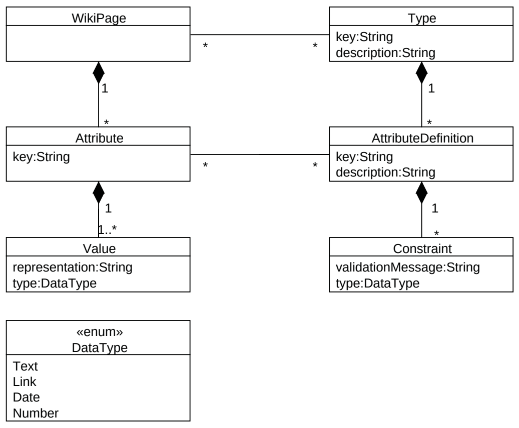


Figure 2: UML class diagram showing Hybrid Wikis core modelling concepts

2.3 Instance and Model Coherence and Evolution

With the possibility to assign attributes to content without using a type in parallel to the creation of models instances can differ from the model. In order to counteract a potential divergence of a model and its instances Hybrid Wikis provide the following mechanisms:

- Suggestions (e.g., of attributes)
- Search for violations of constraints
- Consolidation (e.g., of types)

2.3.1 Suggestions

On the instance level Hybrid Wikis provide attribute and type suggestions. Suggestions are calculated based on page's content and its currently used attributes and types. For instance, if a page is typed *business application* an attribute with key *status* is suggested if another page exists also typed *business application* additionally having an attribute *status*, or a key is suggested if the type of a typed page has an attribute definition.

On the model level, for example, a constraint is suggested if a certain number of pages (i.e., according to a specific threshold) uses similar attribute values. For example, if two pages typed with *business application* provide an attribute *responsible unit* and both have link values to pages typed with *organisational unit* a corresponding constraint is suggested for the attribute definition *responsible unit*, that is, restricting attribute values to be hyperlinks to pages of type *organisational unit*.

Furthermore, all input fields provide autocompletion support in order to guide users towards a consistent usage of terms and model elements. For instance, already used keys are suggested when the user enters an attribute key. Likewise, literals and links are suggested when entering an attribute value.

Suggestions are lightweight means to encourage users to provide structured content without forcing them. Moreover, they help to develop a coherent model when a bottom-up approach is used.

2.3.2 Inconsistency Detection

In Hybrid Wikis structured pages potentially become invalid without modifying the page, for example, by specifying a constraint. Therefore, users are supported in finding inconsistencies. Users can search for wiki pages

- containing any invalid attributes and
- having specific invalid attributes (e.g., searching for pages having an attribute *requester* with invalid values).

Based on these queries users can design their own dashboard in order to be aware of constraint violations.

2.3.3 Consolidation techniques

Once inconsistencies are detected, users can harmonise instances by means of the model. For example, renaming the key (e.g., *status*) of an attribute definition can be propagated to related pages (i.e., attributes of those pages are renamed accordingly). Or constraints can be applied to the related pages. For example, if *responsible unit* is restricted to links to pages with type *organisational unit* related attributes can be transformed into links matching this consistency rule. That is, literals are transformed into links (by creating new pages or referencing existing ones, both extended with type *organisational unit*, if required). In case of links the link target is typed with *organisational unit*, if required.

2.4 Integrated Model Management in Hybrid Wikis

In Hybrid Wikis models emerge from structured wiki pages and hyperlinks between them (i.e., from the actual instances). Model evolution is facilitated by suggesting attribute definitions and constraints based on an analysis of the instances. Evolution on the instance level is facilitated by providing attribute suggestions based on the underlying model. Additionally, the model can be used to harmonise (i.e., consolidate) derivations of the instances, because instances can be constrained by the model. This way, users are urged to follow the schema, but they are never forced.

Figure 3 depicts Hybrid Wikis' instance level and model level and sketches their interplay. Additionally, two participation roles are distinguished (Neubert 2012). Authors collaborate on the instance level by managing wiki pages with attributes and types, while tailors collaborate on the model level by managing attribute definitions and constraints.

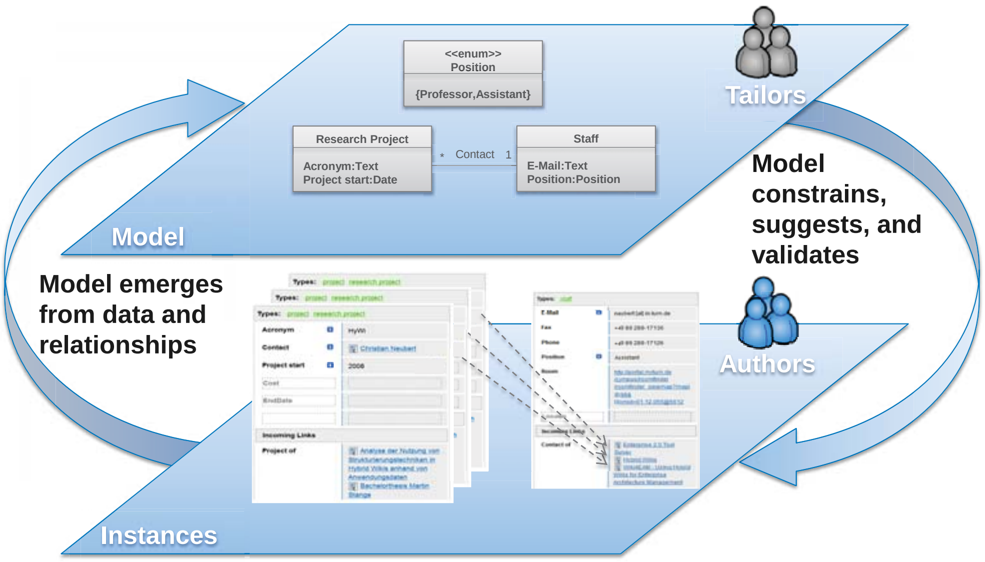


Figure 3: Data-driven information management and schema evolution via collaboration according to Neubert (2012).

2.5 Using Hybrid Wikis for EA Management

Hybrid Wikis provide means to collaboratively manage and model information for different purposes in different application domains such as notes and publications for (personal) information management within an university's chair. In the context of EA management, the idea is to start with existing unstructured information sources captured as wiki pages (e.g., derived from Office documents) and then to incrementally and collaboratively structure these pages with attributes and types as needed for enterprise architecture modelling. In Hybrid Wikis persons providing instance data (i.e., the authors, such as persons from the technical departments having the architectural knowledge) are never prevented from data entry caused by hard schema constraints and structures (i.e., types and attributes) can always flexibly be adapted as needed. This way, the 'true' model emerges bottom-up derived from user-managed, structured wiki pages (i.e., from the instances). Once a core model has emerged, enterprise architects (i.e., the tailors) can make this core explicit (e.g., by means of attribute definitions) and optionally define additional integrity constraints. However, the degree of rigidity can be softened by all users, if needed.

In this article we focus on how to derive EA models from collaboratively created and structured wiki pages and how these models can be visualised. In Buckl et al. (2009a) we discuss further advantages of using wikis for EA management (e.g., versioning, awareness).

3 Visualising Emergent Meta-Models

When authors model collaboratively with Hybrid Wikis, modelling usually occurs implicitly on the instance level. Thus, the model itself is not in the primary focus of participating authors and a visualisation of it is not available upfront. In the following the need for such visualisation as well as the used techniques will be described. In addition, an explicit model can be defined by tailors which also needs to be visualised. Finally, a combination of both is presented.

3.1 The Need for Visualisation

When authors create new instances or adapt the model implicitly during their daily work, they might be interested in its structure. This is particularly the case, if a type has many attributes linking to different other types. In such case, the current way of displaying attributes and their values as a table might not give an adequate overview about a type and its relations to other types. In Hybrid Wikis the authors modelling on the instance level can only see directly associated instances. Hence, they are not able to see the structure or relationships of the associated types. With a UML class diagram visualising the underlying model this drawback of data-driven modelling can be mitigated as it provides a navigational aid. In addition, the visualisation of the model can be used to discover the terminology used on the instance level. This includes, among others, attribute names for new types or similar attributes in different types.

The second target group for the visualisation are enterprise architects who transfer control over the enterprise's model to authors providing data. If doing so, they need a familiar overview about the evolving model to be able to intervene if necessary. For example, if different semantically equivalent terms are used within the model enterprise architects can use the visualisation to discover them. Beside the consolidation of types and attributes enterprise architects have the ability to adapt the model by creating constraints or using the 'strict' property (cf. Sect. 2.2). Attributes declared as strict urge authors to provide a specific model element. Thus, enterprise architects can rather control the model evolution towards a specific target state.

By the use of a UML-like visualisation (cf. Fig. 7), which presents the model defined by the enterprise architects beside the actual instances filled by the authors, the enterprise architect is also able to reflect her decisions forming the enterprise model. With such visualisation at hand, it furthermore becomes easy to discover unused types and attributes, model parts which are regularly contravened as well as 'strict' properties which are not obeyed.

Extending Hybrid Wikis with the capability to visualise the data-driven model in parallel to the explicit model eases its usage for both, authors and tailors.

3.2 Mapping Hybrid Wikis' Instances to UML Object Diagrams

Each instance in Hybrid Wikis can be modeled as a single object in a UML object diagram. Thereby, the page's *type* is used as the UML object's type since it classifies a set of instances having a similar structure. Because there is no obligation to assign a type to each instance, untyped instances will be neglected within the resulting UML diagram.

The *attributes* within Hybrid Wikis can directly be mapped to UML's attributes. Therefore, the attribute's name becomes the UML attribute's name.

Within Hybrid Wikis, *hyperlinks* are the primary means to establish a relationship between two instances. They can either be used in the content section of a wiki page or as an attribute value. As the content part is completely neglected during model derivation, only links used as attribute values are analysed. If the linked instance has a type assigned, a UML association will be derived and the attribute's name will be used as role name. Currently, there is no possibility to define bidirectional links in Hybrid Wikis. Therefore, only independent directed links can be established on the instance level. As consequence, actually bidirectional links are represented as two independent directed associations in the derived UML diagram.

Figure 4 shows an exemplary object diagram. Therein, wiki pages of type *project* are shown with their respective attributes and associations.

As also visualised in Fig. 4 attributes are not associated to the type of an instance with the result that various instances of the same type can have different attributes. Consequently, a respective class diagram has to be abstract enough to account for that kind of diversity.

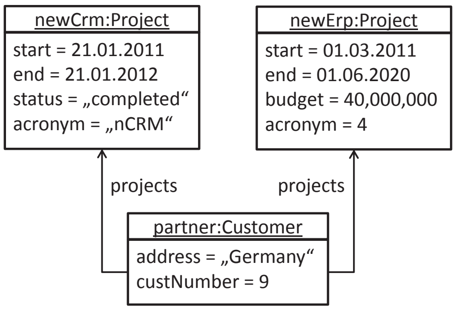


Figure 4: Exemplary UML object diagram of the instances

3.3 Mapping Hybrid Wikis' modelling Elements to UML Class Diagrams

In addition to an object diagram UML class diagrams are better suited to provide a holistic overview about the structure of types used for Hybrid Wikis' instances. The basis for such class diagram forms the already described object diagram. Classes and their attributes as well as their associations can be directly derived from it. By contrast, the attribute's data type has to be derived based on the various instances. Because an attribute's values can have different data types, the model derivation needs to account for inconsistencies. As a preliminary solution, the data type with the highest relative frequency will be used if attribute value types differ between instances.

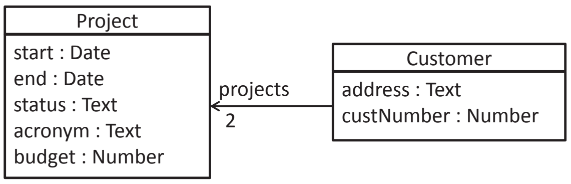


Figure 5: Exemplary UML class diagram of the implicit model

As a result, a class diagram visualising the structure of types used within Hybrid Wikis neglects the fact, that some attributes might not be present for some instances. Based on the object diagram shown in Fig. 4, Fig. 5 shows the corresponding class diagram.

3.4 Explicit Model Visualisation

In parallel to editing instances Hybrid Wikis support modelling also explicitly by the specification of *attribute definitions* and *constraints*. Always bound to a specific type, attribute definitions can be used to explicitly describe and refine type-specific attributes. By the use of constraints the values of an attribute can be defined in more detail. For example, the number of values can be set to 0..1, 1, 1..*, or *. An attribute's data type can be restricted to: Text, Number, Date, or Link whereas for data type Link one can also specify the concrete type the linked instance has to be assigned to. When defining a constraint for a specific attribute the constraint can optionally be marked to be 'strict'. In that case, the constraint will be enforced for all new instances while existing instances will not be changed. Because UML does not account for inconsistencies, the 'strict' property cannot be visualised in standard UML class diagrams. Using the same types already shown in Fig. 5, Fig. 6 shows an exemplary class diagram visualising explicitly modeled types.

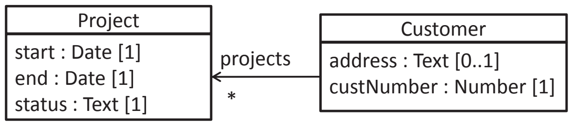


Figure 6: Exemplary UML class diagram of the explicit model

Since the explicit way of modelling is not bound to any instances it can of course lead to types without any instances. In that case, the respective UML classes are still depicted.

3.5 Instance and Explicit Model Co-Visualisation

When deriving UML class diagrams based on both, instance data and explicit model, some characteristics of emergent models cannot be expressed in UML. In the following, suggestions for UML extensions are presented to express emergent model characteristics as well as instance level and model level in parallel. Thereby, each extension is presented in an exemplary UML-like class diagram derived from Hybrid Wiki data used within the domain of project management (cf. Fig. 7). In fact, the previously described class diagram visualising the instance level and the class diagram visualising the explicit model level are thereby combined.

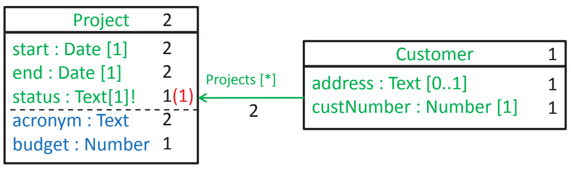


Figure 7: UML-like class diagram visualising an emergent model

3.5.1 Instance versus Model Level modelling

As previously explained, in Hybrid Wikis modelling takes place on two different levels: instance level and model level (cf. Sect. 2). Because UML class diagrams are not intended to distinguish between these two levels, an appropriate mechanism has to be introduced to express the difference of modelling levels.

First, the number of instances is an important characteristic of emergent models. Thus, it has to be reflected within the derived UML-like class diagrams for all elements including classes, attributes, and relationships. This can be achieved by inserting the concrete number after the respective element's name aligned to the right. Thereby, it can be easily distinguished from the UML instance number constraints which use brackets. For example, in Fig. 7 there are two instances of type *project* but only for one instance a *budget* is provided.

Second, within emergent models derived from instance data some information about the underlying model might be missing, because users are not forced to provide it. As a UML class diagram might require information of that kind, it has to be derived based on the available implicit instance data. For example, if all instances of a specific type have an attribute whose value links to exactly one instance of another type, a multiplicity of 1 will be derived for the association. Because the tailors have not already decided about an upper bound for the number of instances, this fact has to be expressed in the resulting diagram. A possible way therefore is the use of colours to express the difference between information provided explicitly by tailors (on the model level) and information derived from the instances. In Fig. 7 below dashed line the colour blue marks elements which are derived from the instances (i.e., the class diagram for instances, Fig 5). For example, the attribute *acronym* is mostly of data type Text but there might also be instances with an attribute *acronym* having another data type. Text is shown in the UML-like class diagram since it has the highest relative frequency.

Third, the distinction between the instance level and model level in general has to be expressed. This includes especially attribute definitions and constraints. Therefore, elements explicitly defined on the model level are highlighted in green and positioned above a dashed line (derived from the explicit class diagram, Fig.6). Using different colours according to the level of modelling directly shows the diagram reader how the respective elements emerged. For example, in Fig. 7 the attributes *start*, *end*, and *status* of type *project* are explicitly defined in the model by the use of attribute definitions and consequently marked green. Especially for associations between two types instance and model data is displayed in parallel if available. For example, the association between the types *project* and *customer* depicted in Fig. 7 shows, that there is a multiplicity constraint of [0..*] explicitly defined in the model but on the instance level each instance of type *customer* has exactly two *projects* assigned.

Within the current visualisation some facts of the instance model are overlaid by the explicit model. If a multiplicity of * is defined in the explicit model for an association the fact that each instance has exactly one object it refers to will be hidden. In addition, for attributes having at least one instance using a hyperlink as value, a UML association is derived even if other instances use other data types like Text. In general, the data type derivation algorithm works as follows: if an attribute definition is present it will be shown accordingly. Otherwise, the data type used most often among the respective instances will be used. If none of this rules returns a single data type the following sequence of types will be used: Link, Date, Number, Text. This sequence ranks hyperlinks highest because we consider knowledge about relationships to be more important than literal attributes. The other data types are ranked according to their specificity.

3.5.2 Model Inconsistencies

The decoupling of the instances from the explicit model in Hybrid Wikis allows for inconsistent instances which do not comply to the constraints defined by the model. Because UML class diagrams disregard possible inconsistencies the expression of that characteristic also requires a UML extension. Again, the concrete number of non-compliant instances is displayed aligned to the right. To distinguish this number from the actual amount of instances, it is put into parentheses. Furthermore, using two different colours, the following two cases of non-compliance can be distinguished:

- The *Instance* violates a *Constraint*
- The *Instance* violates a 'strict' *Constraint*

As strict constraints force users to comply for all newly added instances, this kind of violation might be worse than violating a standard constraint. By using the colours red and yellow, this distinction is visualised. For example, the attribute *status* of type *project* displayed in Fig. 7 has a 'strict' constraint (visualised by an exclamation mark) that its value has to be a Date. As indicated by the red number, one instance is in violation to that constraint, having a *status* attribute of another data type. If the 'strict' property is not set for a specific constraint, a yellow number indicates the number of violations.

4 Application and Evaluation

To validate our approach, in December 2010 at Technische Universität München we established a community, namely Wiki4EAM (Matthes and Neubert 2011), of experienced enterprise architects from 25 large German organisations in order to pursue a lightweight, wiki-based approach to EA management (Buckl et al. 2009a). In the following we introduce the experiences gained in applying Hybrid Wikis in enterprises participating in our Wiki4EAM community. These experiences are described in detail in Neubert (2012).

In the period from December 2010 to March 2012 we conducted seven workshops together with the community members. In the first workshop the participants were introduced to the main concepts underlying Hybrid Wikis by presenting some slides. Additionally, we used a projector to demonstrate the core system functions (e.g., structuring of wiki pages, creating visualisations) by using a small EA management example scenario from the banking industry. After the workshops our software was made available to the members of the community. Some used the system hosted at Technische Universität München, some downloaded it and installed it locally. In the subsequent workshops, members of the community presented their developed wiki-based EA management solutions. Based on the experience gained with the use of Hybrid Wikis, new requirements were collected and discussed. This way, we constantly adapted and improved our solution according to the feedback of our industry partners.

4.1 Survey

In the 6th workshop in December 2011, we asked (paper-based) the community members to provide the main reasons why they are using Hybrid Wikis in their enterprises. The participating members stated (extract and most frequent answers) that

- the model is flexibly adaptable (i.e., can be created incrementally and does not need to be fixed in advance) (6 of 7) and
- Hybrid Wikis are easy to use (i.e., provide a high level of usability and a clean user interfaces) (4 of 7).

Since only seven members attended the community meeting in December 2011, these survey results do not represent a strong, well-founded evaluation. However, the answers allow to assume that the adaptability of structures is the main reason for applying Hybrid Wikis in EA management scenarios. Furthermore, the results indicate that business users understand the concepts underlying Hybrid Wikis since they adapt structures without additional IT-support facilitated by a web interface that participants consider to be easy to use.

4.2 Case Study

In the following we briefly introduce one selected case study from a global operating bank participating in the Wiki4EAM community. This bank is referred to as Bank A subsequently.

4.2.1 Starting Point

In December 2010, Bank A started participating in the community to deepen the knowledge about EA management. In particular, one enterprise architect of a unit concerned with infrastructure architecture management evaluated whether a previously created landscape of the unit's infrastructure can be represented by the concepts provided by Hybrid Wikis. That is, the architect evaluated if it is possible to build a model of the infrastructure landscape consisting of types, attributes, attribute definitions, and constraints. In an additional[1] two hours personal lesson with the architect of the infrastructure unit, the core concepts (e.g., wikis, wiki pages, types, attributes, constraints) were explained in detail and technical questions about some system functions were clarified (e.g., how to define queries, how to embed a query in a wiki page in order to create a visualisation). After this additional lesson, Bank A accessed Hybrid Wikis through a system hosted at TU München. The system included some EA management demo data (e.g., business applications, organisational units) and some exemplary visualisations (e.g., a cluster map showing the relation between organisational unit and business applications).

[1] In addition to the demonstration and visualisation during the initial workshop.

4.2.2 Model

In the period from December 2010 to December 2011 Bank A prototypically modeled the architectural elements of the infrastructure unit in a wiki separated from the wiki with the demo data. The resulting model representing the infrastructure architecture is depicted in Fig. 8.

The concepts of this model represent infrastructure elements on a certain level of detail. The model consists of four levels. Each level represents a different level of granularity in the IT infrastructure stack. For instance, an Infrastructure Component (IC) (level 4) can either be software or hardware. Likewise, an area (level 3) represents the logical unit for a set of ICs (e.g., a set of software components). Bank A incrementally developed this model by using wiki pages structured with attributes and types. The wiki's home page serves as a dashboard. The dashboard shows the relations between some types (e.g., between areas and infrastructure components) either as custom embedded table views or as graphical visualisations (e.g., as a cluster map). Both kinds of customised views answer specific EA management questions. For example, a cluster map shown on this dashboard indicates which ICs are used in which areas, or obsolete ICs are indicated in a (custom) tabular view by filtering for pages (typed with IC) having a colour attribute with value red.

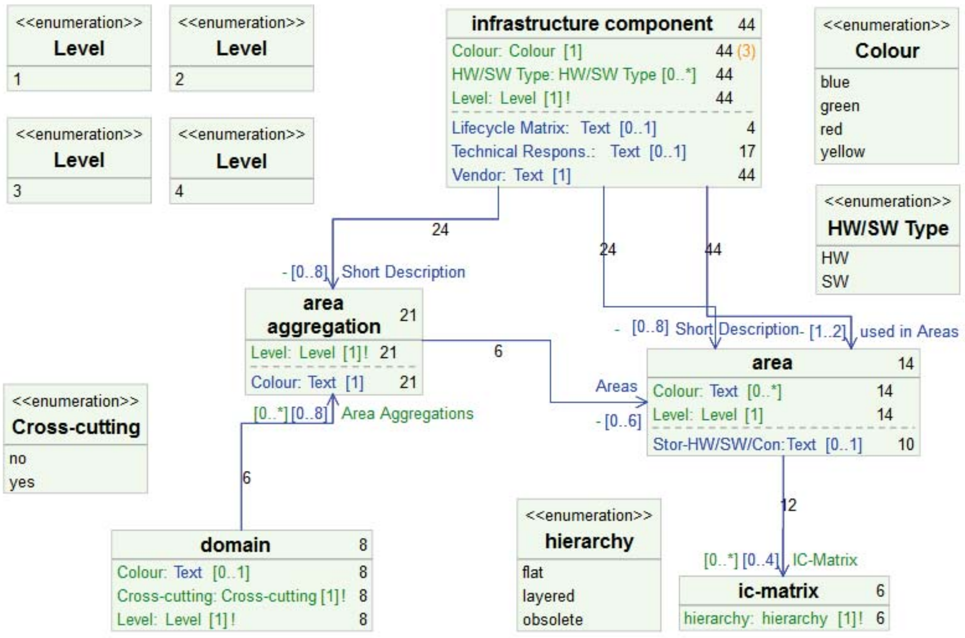

<<enumeration>> Level 1
<<enumeration>> Level 2
<<enumeration>> Level 3
<<enumeration>> Level 4
infrastructure component 44
Colour: Colour [1] 44 (3)
HW/SW Type: HW/SW Type [0..*] 44
Level: Level [1]! 44
Lifecycle Matrix: Text [0..1] 4
Technical Respons.: Text [0..1] 17
Vendor: Text [1] 44
<<enumeration>> Colour
blue
green
red
yellow
<<enumeration>> HW/SW Type
HW
SW
24
- [0..8] Short Description
24
44
area aggregation 21
Level: Level [1]! 21
Colour: Text [1] 21
6
- [0..8] Short Description- [1..2] used in Areas
Areas
area 14
Colour: Text [0..*] 14
Level: Level [1] 14
Stor-HW/SW/Con: Text [0..1] 10
- [0..6]
[0..*] [0..8] Area Aggregations
<<enumeration>> Cross-cutting
no
yes
6
12
domain 8
Colour: Text [0..1] 8
Cross-cutting: Cross-cutting [1]! 8
Level: Level [1]! 8
<<enumeration>> hierarchy
flat
layered
obsolete
[0..*] [0..4] IC-Matrix
ic-matrix 6
hierarchy: hierarchy [1]! 6

Figure 8: Emergent model used by Bank A for the management of infrastructure elements according to Neubert (2012)

4.2.3 Lessons Learned

In April 2011 in a Wiki4EAM community workshop, Bank A summarised the experiences gained with Hybrid Wikis (extract):

- Absolute beginners are not able to start without any introduction making a short lesson (about 2 hours) or user manual mandatory.
- Creating and adapting a model is possible without any further help from IT experts.
- Starting with instances is simpler than creating a holistic model first.
- Having a previously developed model in mind is helpful since it provides at least a basic frame for structuring.
- Gradually adding new attributes to wiki pages works well and creates beneficial structures.

These experiences show that Hybrid Wikis are applicable in the domain of enterprise architecture management. They empowered enterprise architects to develop a model in a data-driven incremental way. Furthermore, the adaptability of the model helped to shape the model according to changing information needs in order to answer typical EA management questions. In the introduced case, it was helpful to have a vague idea of the target model and some demo data at hand.

5 Related Work

In current literature different approaches for enterprise modelling exist. In order to distinguish the approach presented in this article from existing ones prominent representatives of different streams will be described in this section.

5.1 MoKi Wiki

MoKi (Modelling wiKi) is a wiki for the purpose of enterprise modelling (Casagni et al. 2011; Ghidini et al. 2009) built on the basis of Semantic MediaWiki. Its objective is to foster collaborative modelling of the constituents of an enterprise, in particular domain objects, business processes, and competencies. Therefore, templates and forms are used to prevent users from learning a special syntax. By contrast, Hybrid Wikis do not focus on the modelling part; models emerge implicitly while users are documenting knowledge. It is also not tied to a specific domain such as enterprise modelling (Matthes et al. 2011).

5.2 DBpedia

The goal of DBpedia is to extract structured information from Wikipedia's content and to provide it publicly in the Web (Kobilarov et al. 2009; Lehmann et al. 2009). Thereby, a syntactic analysis examines the markup of Wikipedia pages using pattern matching techniques in order to extract RDF statements. More details about the extraction algorithm are described in (Fensel et al. 2011). In Hybrid Wikis, relationships between different pieces of content (wiki pages) can only be created explicitly in the structured part of each page. The unstructured part is not analysed to derive the underlying model up to now.

5.3 EA Management Tools

As outlined in the EA Management Tool Survey 2008 (Matthes et al. 2008) most EA management tools provide a rigid model. Often, such tools also include their own methodology of EA management and are not built to be adapted easily by the customer. Nevertheless, changes of class and attribute names as well as the introduction of new concepts is possible in some tools. By contrast to Hybrid Wikis, such changes cannot be implemented directly by the user but require the interaction of an administrator with special rights. Therefore, changes to a model can be implemented faster in Hybrid Wikis although losing a central governance instance.

5.4 Model and Meta-Model Co-Creation

The parallel evolution of model and meta-model are subject to research, for example, focusing on the model-to-model transformations (Cicchetti et al. 2008; Ruscio et al. 2011). Such approaches analyse possibilities of propagating changes made to a meta-model to the model and vice-versa. By contrast, Hybrid Wikis do not apply changes to one model to the other. Instead Hybrid Wikis just highlight possible mismatches of model and meta-model. Furthermore, both layers of models can now be visualised together in order to show implicit and explicit information structures in parallel.

5.5 Visualisation of Models

The visualisation of data models has a long tradition. In order to visualise a model it has to be present in some kind of standardised format. For instance, a widely used meta-model to describe models named Ecore has been developed by the Eclipse community (Steinberg et al. 2004). Because Ecore can be used to describe models as well as meta-models (since they are also models) both can be visualised by many different tools. But the goal of Hybrid Wikis is to visualise both models in a single diagram. To the authors' knowledge there is no standardised format nor tool available to create a visualisation depicting both instances and their underlying model.

6 Conclusion

In this article we discussed how the concept of Hybrid Wikis fosters the collaborative and integrated (meta-) model development in the context of EA management. In Sect. 2, we briefly introduced the modelling concepts of Hybrid Wikis and explained how their provided mechanisms facilitate (meta-) model evolution. Subsequently, in Sect. 3, we presented a new visualisation of emergent models inspired by UML class diagrams and enriched by additional information from the instance level. To validate our approach, in Sect. 4, we presented the results of a survey conducted among members of our Wiki4EAM community who applyed Hybrid Wikis within their organisation. Additionally, we presented a case study in a global operating bank using Hybrid Wikis for infrastructure modelling. We compared Hybrid Wikis in Sect. 5 with other modelling approaches described in scientific literature.

7 Discussion and Future Research

Although Hybrid Wikis seem to be useful for enterprise modelling their usage is subject to some preconditions. If used for EA management, the implementing organisation has to shift its culture to an open corporate culture. Enterprise architects transfer part of their control over the enterprise's model to employees responsible for data input. Since the presented collaborative enterprise modelling approach depends heavily on the active contribution of the employees having the architectural knowledge data input has sometimes to be motivated, for example, by architects leading with good example. Another lesson learned during the evaluation was that starting modelling from scratch is difficult. Therefore, Hybrid Wikis need to provide the ability to import existing modelling solutions and additionally to adapt them. Another threat of using Hybrid Wikis for enterprise modelling are subsequent alternations of versions also known as edit wars, cf. Viégas et al. (2004). The presented novel approach to visualise instances in parallel to their explicitly defined model has still some limitations. Possibly, mutually directed associations are used instead of bidirectional associations. Moreover, the visualisation readers might be mislead by an attribute data type derived from heterogeneous instance data and the number of its instantiations. If different data types are used for a single attribute, the actual visualisation suggests that all instances have the same data type.

In addition, future research should a) examine if Hybrid Wikis can serve as model repository to exchange and merge models from different EAM scenarios, b) investigate possibilities to support behaviour and views based on these emergent, adaptive models, and c) analyse and compare models of many other Wiki4EAM community members, for example, to identify modelling patterns.

References

Azevedo C. L. B., Almeida J. P. A., van Sinderen M., Quartel D., Guizzardi G. (2011) An Ontology-Based Semantics for the Motivation Extension to ArchiMate. In: Enterprise Distributed Object Computing Conference (EDOC). Helsinki, pp. 25–34

Buckl S., Matthes F., Neubert C., Schweda C. M. (2009a) A Wiki-based Approach to Enterprise Architecture Documentation and Analysis. In: The 17th European Conference on Information Systems. Verona, Italy, pp. 2192–2204

Buckl S., Matthes F., Schweda C. M. (2009b) Future Research Topics in Enterprise Architecture Management - A Knowledge Management Perspective. In: Workshop Trends in Enterprise Architecture Research (TEAR 2009). Stockholm, pp. 1–11

Bunge M. (1977) Treatise on Basic Philosophy: Ontology I: The furniture of the world. Reidel Publishing, New York

Casagni C., Francescomarino C. D., Dragoni M., Fiorentini L., Franci L., Gerosa M., Ghidini C., Rizzoli F., Rospocher M., Rovella A., Serafini L., Sparaco S., Tabarroni A. (2011) Wiki-Based Conceptual Modeling: An Experience with the Public Administration. In: Aroyo L., Welty C., Alani H., Taylor J., Bernstein A., Kagal L., Noy N. F., Blomqvist E. (eds.) International Semantic Web Conference (2). Lecture Notes in Computer Science Vol. 7032. Springer, pp. 17–32

Cicchetti A., Ruscio D., Eramo R., Pierantonio A. (2008) Automating co-evolution in model-driven engineering. In: Ceballos S. (ed.) 12th International Enterprise Distributed Object Computing Conference (EDOC), IEEE Computer Society

Fensel D., Facca F. M., Simperl E., Toma I., Fensel D., Facca F. M., Simperl E., Toma I. (2011) Semantic Web. In: Semantic Web Services. Springer, pp. 87–104

Gerber A., Kotzé P., van der Merwe A. (2010) Towards the Formalisation of the TOGAF Content Metamodel Using Ontologies. In: Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS), pp. 54–64

Ghidini C., Kump B., Lindstaedt S., Mahbub N., Pammer V., Rospocher M., Serafini L. (2009) MoKi: The Enterprise Modelling Wiki. In: Aroyo L., Traverso P., Ciravegna F., Cimiano P., Heath T., Hyvönen E., Mizoguchi R., Oren E., Sabou M., Simperl E. (eds.) The Semantic Web: Research and Applications Vol. 5554. Springer, Berlin, pp. 831–835

Kobilarov G., Bizer C., Auer S., Lehmann J. (2009) DBpedia - A Linked Data Hub and Data Source for Web Applications and Enterprises. In: Proceedings of Developers Track of 18th International World Wide Web Conference (WWW 2009). Madrid, Spain

Källgren A., Ullberg J., Johnson P. (2009) A Method for Constructing a Company Specific Enterprise Architecture Model Framework. In: 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing. IEEE, pp. 346–351

Lehmann J., Bizer C., Kobilarov G., Auer S., Becker C., Cyganiak R., Hellmann S. (2009) DBpedia - A Crystallization Point for the Web of Data. In: Journal of Web Semantics 7(3), pp. 154–165

Lucke C., Krell S., Lechner U. (2010) Critical Issues in Enterprise Architecting - A Literature Review. In: Proceedings of the Sixteenth Americas Conference on Information Systems. Lima, Peru

Matthes F., Neubert C. (2011) Wiki4EAM: Using Hybrid Wikis for Enterprise Architecture Management. In: 7th International Symposium on Wikis and Open Collaboration (WikiSym). Mountain View, California, USA

Matthes F., Buck S., Leitl J., Schweda C. M. (2008) Enterprise Architecture Management Tool Survey 2008. TU München, Chair for Informatics 19, Prof. Matthes (sebis)

Matthes F., Neubert C., Steinhoff A. (2011) Hybrid Wikis: Empowering Users to Collaboratively Structure Information. In: Proceedings of the 6th International Conference on Software and Data Technologies. Seville, Spain, pp. 250–259

Neubert C. (2012) Facilitating Emergent and Adaptive Information Structures in Enterprise 2.0 Platforms (to appear). PhD thesis, Fakultät für Informatik, Technische Universität München

van der Raadt B., Schouten S., van Vliet H. (2008) Stakeholder Perception of Enterprise Architecture In: ECSA 2008 Morrison R, Balasubramaniam D, Falkner K (eds.) Springer, pp. 19–34

Ruscio D., Ralf L, Pierantonio A. (2011) Automated co-evolution of GMF editor models. In: Software Language Engineering. Springer, pp. 143–162

Steinberg D., Budinsky F., Merks E., Paternostro M. (2004) EMF: Eclipse Modeling Framework, Second Edition

Viégas F. B., Wattenberg M., Dave K. (2004) Studying cooperation and conflict between authors with history flow visualizations. In: Proceedings of the 2004 Conference on Human Factors in Computing Systems - CHI ’04 6(1), pp. 575–582

Zachman J. A. (1987) A framework for information systems architecture. In: IBM Systems Journal 26 (3), pp. 276–292

Florian Matthes, Christian Neubert, Alexander W. Schneider

Chair for Software Engineering for Business Information Systems (sebis),
Institute for Informatics,
Technische Universität München
{matthes | christian.neubert | alexander.schneider}@tum.de