Lessons Learned in Aligning Data and Model Evolution in Collaborative Information Systems

Thomas Reschenhofer
Technical University of Munich (TUM)
Munich, Germany
reschenh@in.tum.de

Manoj Bhat
Technical University of Munich (TUM)
Munich, Germany
manoj.mahabaleshwar@tum.de

Adrian Hernandez-Mendez
Technical University of Munich (TUM)
Munich, Germany
adrian.hernandez@tum.de

Florian Matthes
Technical University of Munich (TUM)
Munich, Germany
matthes@in.tum.de

ABSTRACT

Today's enterprises have to align their information systems continuously with their dynamic business and IT environment. Collaborative information systems address this challenge by involving diverse users in managing the application's data as well as its conceptual model. In this sense, both the data and the model co-evolve. There are different approaches for aligning data and model evolution, wherein either the data is aligned to the model, or vice versa.

In this work, we present a hybrid approach supporting both strategies and elaborate on our experiences of applying the approach in projects for over five years. Thereby, we discuss challenges and issues faced in those projects, and how we addressed them by redesigning and reimplementing the approach. We formulate those issues and respective solutions as lessons learned, which not only hold for the concrete system which was applied in those projects, but which should guide the design and implementation of all software systems supporting the co-evolution of data and model.

Keywords

Lessons learned, best practices, model evolution, data evolution, collaborative information systems, semantic wiki

1. INTRODUCTION

The demand and requirements for information systems are changing continuously due to an increasingly turbulent business environment, technology innovations, and legal regulations [1]. Adaptive information systems enable enterprises to adapt their software systems to meet the demands of such a dynamic business and IT environment. This enables an enterprise to focus on its business and to quickly setup and update its information systems without investing too much time and effort (and thus money) on the technology that supports its business [25, 36, 29].

One aspect of an information systems that is subject to frequent changes is its conceptual model—also referred to as the user-model [22]. The reasons for the user-model changes are manifold, and range from the correction of mistakes to the adaption to new laws and regulations. If the information system is not able to adapt to the changing environment, the quality of the system's support for its business will decrease over time [35]. Therefore, meta-model based information systems that allow users to dynamically update and evolve their user-models in order to meet the demands of the changing business needs are becoming popular. In a collaborative environment, this approach implies at least two different co-existing user roles for managing the user-model and its application data [34, 37], namely model designers who are responsible for user-model changes, and end-users or data owners performing data changes.

The traditional top-down approach to model evolution where user-models evolve and the existing data is migrated to its newer model version has been documented extensively [3, 23, 28, 32]. On the other hand, the popularity of agile methodologies in software engineering and the advent of big data suggests a bottom-up approach where the data is captured first and then the models are extracted or refined based on the data. The bottom-up approach to modeling not only supports quick prototyping and iterative modeling but also allows end-users to capture the data without being restricted by the pre-defined user-model [20]. Subsequently, model designers are notified regarding the availability of new structures or patterns in the data that needs to be considered in the user-model so that it can be adapted according to the evolving data. We refer to such a collaborative approach as the co-evolution of the user-model and its data.

Achieving such a collaborative environment that supports the evolution of both the user-model and its data in a coherent and consistent manner is a non-trivial task. Matthes et al. [20] tackle this challenge with the so-called Hybrid Wiki approach (c.f. Section 2.2). Thereby, the application data is initially represented by the unstructured wiki pages which can be structured incrementally and collaboratively by attaching types, attributes, and integrity rules. At the same time, model designers can define and adapt the user-model which imposes certain constraints on the underlying wiki pages and thus induces a schema on the application data.

ICSE '16 Companion, May 14-22, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4205-6/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2889160.2889235

However, the application of a collaborative information system (CIS) implementing the Hybrid Wiki approach in a variety of use-cases and domains—e.g., Enterprise Architecture Management [19, 5] and Collaborative Product Development [29, 12]—revealed a couple of challenges and issues which are not only related to the Hybrid Wiki approach in particular, but to collaborative approaches for the co-evolution of user-models and data in general. In this paper we discuss our experiences of applying the Hybrid Wiki approach in industrial and research projects, and the consequences for the redesign of this approach. In this sense, we answer the following research questions:

1. What are the challenges in collaborative approaches for managing the co-evolution of the user-model and its corresponding data?
2. How can a CIS address those challenges?

This work can be understood as iterations in the Design Science approach as defined by Hevner et al. [14]. Thereby, the Hybrid Wiki approach as the corresponding design artifact was evaluated by its application in multiple use-cases and subsequently redesigned according to the respective findings. In this context, those findings refer to issues faced in the use-cases as well as solutions and suggestions addressing those issues. Therefore, the main contribution of this paper and thus the additions to the knowledge base in the area of collaborative and model-based information management are the lessons learned [27] in multiple applications of a CIS supporting the alignment of data and model evolution.

Based on the applied research methodology, the remainder of this paper is organized as follows: Section 2 presents the concepts forming the foundations for the main contribution of this paper, which includes a detailed description of the Hybrid Wiki approach. Thereafter, Section 3 summarizes the cases of applications of the CIS and thus the results of the evaluation as part of a Design Science iteration. Section 4 details a list of lessons learned originating from the aforementioned applications. Thereby, these lessons not only describe the challenges but also suggest how a CIS can address them. Therefore, this section provides answers to the aforementioned research questions. While Section 5 gives an overview of the related work, Section 6 concludes the present work and suggests future research possibilities.

2. FOUNDATIONS

In this section, we will first discuss the top-down and bottom-up approach to modeling and set the context of the evolution of the user-model and its data. We then present the Hybrid Wiki approach that supports the evolution of both the model and its data in a collaborative environment.

2.1 Evolution of Model and Data in Collaborative Information Systems

As discussed in Section 1, a CIS that supports the collaborative evolution of the user-model and its data provides appropriate constructs for two different user roles, namely model designers and data owners. The former captures the user-model which forms the core of that system, while the latter is responsible for capturing instances of the concepts in the user-model, their attributes, and their relationships.

2.1.1 Top-down Approach to Modeling (Model-first)

In the first iteration of the model-first approach, the model designer defines the user-model based on the requirements. Using this user-model, system developers realize the model by implementing concepts, relationships, and their constraints. Alternatively, system developers generate the code semi-automatically for the system using the Model-Driven Development approach [2]. Finally, the implemented system is deployed and made available to the data owners to capture the data that is consistent with the predefined model. This process results in context-dependent systems which require updating the system at code-level to accommodate changes in the user-model [22]. Furthermore, it also requires appropriate data transformation mechanisms to migrate the existing data so that it complies with the updated user-model.

The existing approaches that focus on automatic data transformation are restrictive in the sense that they ensure the consistency of the existing data with the updated model. For instance, to ensure a restrictive multiplicity relationship (one-to-many relationship is updated to exactly one relationship) a predefined rule such as "delete all relationships except the first one" is executed to ensure that the existing data is consistent with the new model. Defining such rules when evolving user-models can result in information loss in the CIS. A similar analogy referred to as "schema on write" can be found in data warehousing systems where the schema is defined in databases, and the extract, transform, and load approach ensures that the data is transformed to comply to the schema before it is persisted [9].

To summarize, the model-first approach (a) structures the problem domain using concepts and relationships, (b) provides a consistent view over the data, and (c) restricts users from adding inconsistent data that does not confirm to the defined user-model.

2.1.2 Bottom-up Approach to Modeling (Data-first)

One of the requirements of CIS is their interoperability with other systems. They should support the collection of voluminous and high velocity data (big data) from different sources. This becomes a challenge in the traditional model-first approach as transformation engines need to be implemented to ensure that the incoming data fits the predefined user-model. Referring back to the analogy of data warehousing systems, this problem is addressed by collecting all the incoming data streams ("schema on read") and then providing the flexibility for applications to decide what information they need and in which format [9].

Furthermore, typically in green-field projects that follow an agile development approach, the requirements become clear after multiple iterations of software development phases. In model-based information systems, new requirements typically involve improvements of the user-model as data owners are unable to capture the data using the existing user-model. The data-first approach provides the flexibility to data owners to capture their demands for instance by adding a custom attribute to an entity. On receiving notifications regarding the same, the model designer can decide if to incorporate them as part of the user-model or not, i.e., if the newly added attribute is generic enough then a corresponding property can be created for the relevant concept.

The benefits of the data-first approach are that it (a) provides the flexibility to data owners to contribute to the user-model definition, (b) reduces the number of hand-shakes required in the process of a user-model update, and (c) facilitates the integration of data from multiple data sources.

However, implementing only the data-first approach could also be disastrous in information systems, as the system would not support the much-required benefits of the model-first approach. Thus, conscious trade-offs must be made while developing a CIS that supports both the model- and data-first approaches.

2.2 The Hybrid Wiki Approach

The goal of the Hybrid Wiki approach is to empower non-expert users to collaboratively gather and consolidate information in a knowledge-based information system [20]. It tries to reduce the complexities involved in using semantic wikis and their corresponding technologies including the markup language and the query language. The term "hybrid" refers to wiki pages which integrate a subset of semantic wiki features into classical wiki software.

The core concepts *attribute* and *type tag* in the Hybrid Wiki meta-model as shown in Figure 1 allow users to capture the knowledge in a semi-structured *wiki page*. An attribute is a key-value pair associated with a *wiki page*. The attribute has a name and potentially multiple values of different types, e.g., strings or links to other wiki pages. A data owner can create an attribute at run-time to capture structured information about a wiki page. A type tag allows users to refer to a collection of similar pages, e.g., organizations, projects, etc. A data owner can associate multiple *type tags* with a *wiki page*. These concepts structure the data in a wiki page and capture the data in a data-first manner as discussed in Section 2.1.2.

Furthermore, the concepts *type tag definition*, *attribute definition*, and *validator* specify constraints on the data. The type tag definition and attribute definition are loosely coupled with type tags and attributes respectively through their name. The type tag definition consists of multiple attribute definitions which in turn contain multiple validators such as *multiplicity validator*, *string value validator*, and *link value validator*. A wiki page associated with a type tag definition through a type tag may therefore contain attributes corresponding to attribute definitions. Furthermore, an attribute and its values can be associated with validators for maintaining integrity constraints. These mappings, which are also indicated in Figure 1 with the dotted lines, enable model designers to specify soft constraints on the wiki pages and its attributes. The model designer defines the user-model using type tag definition, attribute definition, and validators, and urges the data owners to capture data corresponding to the user-model. This corresponds to the model-first approach as discussed in Section 2.1.1. However, it should be noted that in the Hybrid Wiki approach data owners are not restrained by strict integrity constraints while capturing information in wiki pages and their attributes, i.e., a value violating integrity constraints as defined in the user-model can still be stored in the CIS [20].

Let us consider the implementation of an exemplary use-case of a Publication Management System (PMS) for managing publications in a conference using the Hybrid Wiki approach. Initially, concepts such as conference and publication are not yet defined as type tag definitions. The data owners responsible for capturing the publications in a conference start to document the knowledge in wiki pages. As the number of publications increases over time, patterns

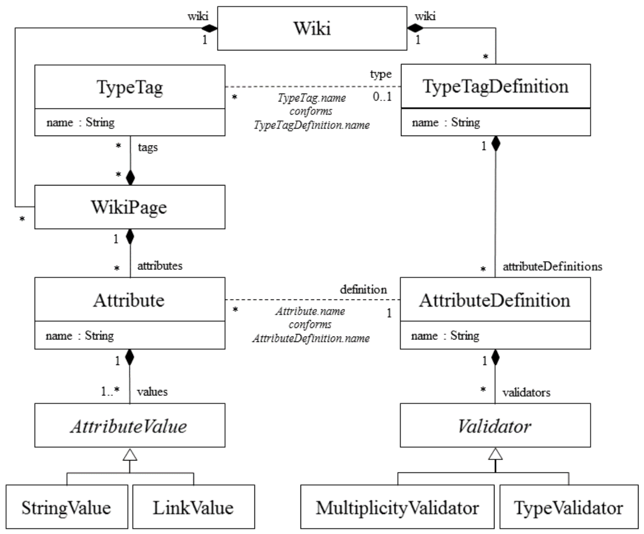


Figure 1: The Hybrid Wiki meta-model [20]

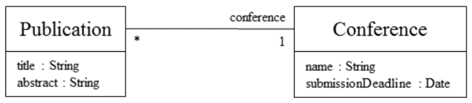


Figure 2: An exemplary user-model of a PMS

in the attributes of the wiki pages representing publication emerge over time, e.g., title, abstract, and authors. Based on those patterns, the model designer creates a corresponding type tag definition *Publication*, which is defined through its attribute definitions along with cardinality constraints (c.f., Figure 2). The wiki pages annotated with the Publication type tag that do not conform to the soft integrity constraints are highlighted in a table so that they can be harmonized by data owners. Consequently, when creating a new wiki page of type Publication, the data owner is notified to provide its associated attributes. If a required attribute is not available, new attributes can be created for wiki pages, and the process is reiterated. This end-to-end example illustrates how the Hybrid Wiki approach supports the co-evolution of the user-model and its data in a collaborative way.

3. APPLICATIONS OF HYBRID WIKIS

In this section we present our experiences in applying the Hybrid Wiki approach in several industrial and research projects. We classify these projects into Enterprise Architecture Management (EAM), Collaborative Product Development (CPD), and Collaborative Content Management (CCM) and present a few selected facts for each category.

3.1 Enterprise Architecture Management

The so-called Wiki4EAM community was started in 2010 by researchers at the Technical University of Munich (TUM) in collaboration with 25 large German enterprises to apply a lightweight approach to EAM using the Hybrid Wiki approach [19]. The successful projects discussed in [5, 18, 20, 33] were conducted as part of the Wiki4EAM initiative to document the current state of the EA by considering viewpoints of different stakeholders in the enterprises. The

Hybrid Wiki approach supported the stakeholders in documenting specific parts of the EA to form a holistic view of the enterprise. The outcomes from the related projects indicated the following positive aspects of using the Hybrid Wiki approach for documenting the EA:

- Stakeholders are empowered to collaboratively revel their information demand.
- Through the collaborative documentation of the EA model in Hybrid Wikis, the EA model emerges *bottom-up* in a very short period of time.
- Documentation is created in the early stages of an EA initiative.
- Scenarios in which the target model is not completely known are supported.
- The evolution of EA models with changing information demands at run-time is facilitated.

Some of the challenges as indicated by the collaborating industry partners include:

- A lack of strict integrity constraints results in limited querying capabilities.
- The modeling capabilities in the Hybrid Wiki approach are restricted. For example, it does not support the explicit definition of type-subtype relationships.

3.2 Collaborative Product Development

The application of the Hybrid Wiki approach in a cooperative new product development (NPD) project (named SmartNets) enabled collaborating networks of small and medium enterprises (SMEs) to develop new products, processes, and services [12, 17, 29]. This research project was funded by the European commission and started in April 2011 involving 15 companies from different European countries. The SmartNets project not only used the Hybrid Wiki system for knowledge management but also for defining a collaboration model for the involved SMEs to develop knowledge-intensive products. The SmartNets collaboration model was first modeled using the Hybrid Wiki approach in a top-down manner. The partners in the industrial network of the project documented their development activities thereafter. The artifacts of the collaboration model include *development project*, *development phase*, *activity type*, *task*, *meeting* and *results* which are modeled as *type tag definitions* in the Hybrid Wiki system (c.f. [12]). The data owners capture these project-specific artifacts as instances of the collaboration model enabling a visual representation of the progress of the project within specific visualizations and dashboards. For example, the so-called *SmartNets Navigator* [17, 30] provides recommendations and support functions for project partners, e.g., which activity should be performed in a specific development phase, what is the relevance of an activity to the current state of the project, and which experts could help when performing a specific activity.

The application of the Hybrid Wiki approach in the SmartNets project helped to improve the system both from the point of functionality as well as from the UI perspective as discussed in Section 4. While defining the collaboration model during the early phase of the SmartNets project, it was possible to adapt and update the collaboration model to meet the needs of all the collaborating network partners. Furthermore, the off-the-shelf features of the Hybrid Wiki system such as the users and group management, notification of events, version management, and document management enabled the SmartNets project initiative to quickly set up the required IT infrastructure and to focus on the project.

3.3 Collaborative Content Management

The Hybrid wiki approach has been extensively used for managing the content of organizations in various contexts, including product management [12], intranets[1], issue management [19], and social event management[2]. In all these projects the *top-down* approach was applied where an initial user-model was implemented based on the initial set of requirements from the customers. It should be noted that in these projects the target model could only be partially derived from the initial set of requirements. However, domain experts were able to adapt the user-model of the domain accordingly. Furthermore, there was a clear separation between the roles of model designers and data owners. The end users of the above projects are the data owners who focus on gathering and analyzing the information. The positive feedback from them include:

- The initial requirements are quickly implemented and the prototype is available for testing and for improving the requirements.
- Change requests related to the user-model are easily incorporated.
- Features including alerts, e-mail notifications, version control, and user management are readily available out-of-the-box.
- The spreadsheet-like UI to capture records of instances of a specific type is user-friendly. Non-technical end users do not require extensive training.

The critical feedback indicating the need for improvement in the Hybrid Wiki approach includes:

- Time and effort is required for customizing the system to meet the UI requirements for individual customers.
- Separate views for model designers and data owners are required, wherein the views for data owners should be simple and intuitive.

4. LESSONS LEARNED IN ALIGNING DATA AND MODEL EVOLUTION

Patton [27] defines lessons learned as the knowledge which is derived from the screening of a situation and which can be applied in similar situations in the future. He also defines the criteria for generating high-quality lessons and formulates them as questions, whose answers in the context of this paper are described in Table 1.

We distinguish between two types of lessons learned, namely lessons about (a) how the model of the Hybrid Wiki approach is adapted and (b) how the Hybrid Wiki system is presented to the users. In the subsequent subsections we relate each of the lessons learned to the concrete use-cases and applications of the Hybrid Wiki approach as discussed in Section 3.

[1]intranet.in.tum.de, wwwmatthes.in.tum.de
[2]informatik-studieren.de

What is meant by a ”lesson”?	An issue we faced in one of the applications of the Hybrid Wiki approach as described in Section 3.
What is meant by ”learned”?	Discussion of identified issues and redesign as well as reimplementation of the Hybrid Wiki approach.
By whom were the lessons learned?	By participants of regular workshops for discussing the application of the Hybrid Wiki system in the respective domain [19], or by a specific user role which is directly involved in the Hybrid Wiki system's application, e.g., the facilitator in CPD [29].
What's the evidence supporting each lesson?	Multiple applications of the Hybrid Wiki approach in different domains with a variety of users in a time-span of more than five years.
What's the evidence the lessons were learned?	Redesign and reimplementation of the Hybrid Wiki approach as well as its successful application in new research and industry projects.
What are the contextual boundaries around the lessons?	The lessons apply to model-based collaborative information systems supporting both the data-first and the model-first approaches.
Are the lessons specific, substantive, and meaningful enough to guide practice in some concrete way?	The lessons are mapped to concrete conceptual and technical adaptions of the Hybrid Wiki system.
Who else is likely to care about these lessons?	Every researcher or practitioner designing or developing a CIS supporting the co-evolution of data and model.
What evidence will they want to see?	Section 3 gives an overview over research and industrial projects where the Hybrid Wiki approach was applied.
How do these lessons connect with other lessons?	The lessons described in the following are mostly connected to each other. Additionally, Section 5 discusses related research approaches and their similarities and differences compared to the lessons learned as presented in the current work.

Table 1: Questions for generating high-quality lessons learned by Patton [27], and their answers regarding the lessons learned as presented in the current work.

4.1 Concept-related Lessons Learned

In this section, we present the changes made to the initial Hybrid Wiki meta-model. Those changes are founded on the goal of simplifying the adoption process in industry.

4.1.1 Change in Terminology

The Hybrid Wiki approach was motivated in the context of Enterprise 2.0 [21] as an extension to the concept of wikis for lightweight collaborative knowledge management. Consequently, the starting point for defining the terminology of the Hybrid Wiki meta-model are wikis and wiki pages. However, these terms already refer to a certain form of representation and content creation.

As previously discussed in Section 3, the Hybrid Wiki approach was not only applied as a means for traditional knowledge management but also as an user-driven and model-based repository, e.g., for capturing architecture elements in an EAM project or tasks and processes in a CPD project. Consequently, the stakeholders in the respective cases refer to their information objects as *Entities* instead of wiki pages, and *Workspaces* instead of wikis. Furthermore, it turned out that the model designers rather used the term *EntityType* instead of *TypeTagDefinition*. Therefore, in order to foster the adoption of the Hybrid Wiki approach in the future projects, we applied those implicitly proposed terminology changes to the Hybrid Wiki meta-model.

Figure 3 shows the adapted Hybrid Wiki meta-model. The concept of workspaces is a means for clustering and separating user-models—defined by entity types and attribute definitions—with their data—constituted by entities and attributes. This means that each entity type and each entity is contained in one specific workspace. However, the Hybrid Wiki approach also allows cross-workspaces relations between entities. The semantics of the concepts of the Hybrid Wiki meta-model in Figure 3 are still the same as for the respective counterparts of the initial meta-model as described in Section 2.2. Further changes of the meta-model's terminology are discussed in Section 4.1.5.

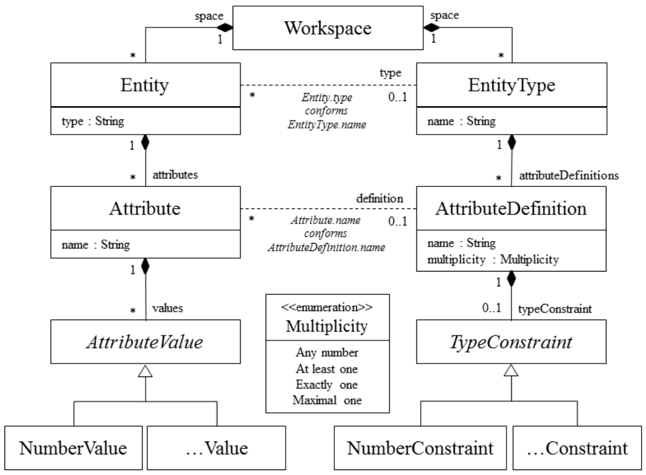


Figure 3: The updated version of the Hybrid Wiki meta-model.

4.1.2 Redesign of Relation from Entity to Entity Type

In the earlier version of the Hybrid Wiki meta-model, a wiki page could belong to multiple type tag definitions through the concept of type tags. This allowed users to relate a wiki page to more than one type tag definition. Furthermore, model designers were able to create generalization relationships among type tag definitions. Even though these features were successfully applied in collaborative content management projects, stakeholders in the domains of EAM and CPD reported that this increased the complexity of the solution. Furthermore, and as also discussed by Matthes et al. [20], one of the challenges of using generalization is to manually ensure that each page with a specific type tag is also assigned to the more general type tag.

To ensure that the Hybrid Wiki system is intuitive and self-explanatory from an end-user's perspective and manageable from a model designer's perspective, we decided to reduce the complexity of the Hybrid Wiki meta-model by simplifying the coupling between entities and entity types. In the Hybrid Wiki meta-model, an entity can be associated with only one type. The dotted lines between concepts in the Hybrid Wiki meta-model as shown in Figure 3 indicate a loose coupling between the respective concepts, i.e., they are related based on the equality of their *type* and *name* properties respectively. For example, an entity is associated with a type whose name is equal to the entity's type property. Therefore, by removing the concept of type tags, entities and entity types are now directly connected to each other. Analogous to this, an attribute is associated with an attribute definition of the respective entity's type through their name. Keeping in mind that the Hybrid Wiki approach supports both the model-first and data first-approach, data owners can still create entities without a predefined entity type. Similarly, data owners can also create attributes that are not associated with a corresponding attribute definition.

4.1.3 Free Attributes

Irrespective of the application domain, the ability to unfold the user-model of its application domain through the involvement of end-users for capturing knowledge is one of the most appreciated features of the Hybrid Wiki approach. In the initial version of the Hybrid Wiki system, the creation and management of wiki pages was supported by attribute suggestions from the system, which were generated based on explicitly defined attribute definitions. For example, if a type tag definition contains an attribute definition, wiki pages having a corresponding type tag are assumed to have a respective attribute. Furthermore, the similarity of wiki pages was used for generating suggestions for attributes. This means that a wiki page is likely to have the same attributes as similar ones. In this context, similarity between wiki pages is determined through their type tags, i.e., the more type tags they share, the more similar they are.

In the updated version of the Hybrid Wiki meta-model, the reduction of the complexity of the relationship between entities and entity types by discarding the type tag concept results in a rapprochement of the user-model and its data, i.e., from a conceptual perspective, the relationship between entities and entity types is more direct, since there is no intermediary concept *type tag* anymore. The same is true for attributes and corresponding attribute definitions. As a consequence, this emphasizes the distinction between those attributes which have a corresponding attribute definition, and those which do not. With respect to this distinction, the notion of *free attributes* referring to attributes without corresponding attribute definitions established itself in discussions with users of the Hybrid Wiki system.

4.1.4 Inverse Role Names

A main feature of the Hybrid Wiki approach is the discoverability of semantic relations between entities through inverse links, i.e., each entity knows which entities are referring to it. For example, in the PMS example in Section 3, there is a relation *conference* from entity type *Publication* to *Conference*, i.e., each publication refers to the conference in which it was published. At the same time, the Hybrid Wiki system generates an inverse relation following a simple naming pattern, namely "{role name} of". For the PMS example, the inverse relation from conferences to publications would be named *conference of*.

However, as observed in many projects of different domains (c.f., Section 3), this name generation pattern does not always produce meaningful names, in particular if users are using verbs for role names instead of nouns, e.g., "published in" instead of "conference" in the PMS example, which would result in "published in of". Furthermore, using a language other than English for the definition of the user-model also yields to undesirable names for inverse roles. In order to address this issue, we extended the Hybrid Wiki meta-model and in particular the attribute definitions concept with the property *Inverse Role Name* enabling model designers to take control of the naming of inverse roles.

4.1.5 Extension of Constraints and Validators

The initial Hybrid Wiki meta-model supported two types of attribute values: *TextValue*s and *LinkValue*s. Thereby, end-users were able to attach either textual information to wiki pages, or to relate them with other wiki pages through links. At the same time, model designers could refine attribute definitions by creating *MultiplicityValidator*s and *TypeValidator*s. The multiplicty validators constraint the number of values for the respective attributes (e.g., "exactly one value", or "at least one value"), whereas type validators specify the set of valid types for corresponding attributes. Initially the Hybrid Wiki meta-model only supported type validators for links, texts, and enumerations. The enumeration validator enforces specific *TextValue*s for corresponding attributes, while the type validator can be further refined to make sure that corresponding attributes only refer to wiki pages with a certain type tag. Referring again to the PMS example in Figure 2, a model designer could define an attribute definition *conference* for the type definition *Publication*. Thereby, multiplicity and type validators can be specified to ensure that each publication must have exactly one *LinkValue* to a wiki page with the type tag *Conference*.

However, the application of the Hybrid Wiki system in different domains—particularly in EAM—revealed that more attribute types are required, e.g., *Number*, *Date*, and *Boolean*. Therefore, we extended the Hybrid Wiki meta-model by respective specializations of *AttributeValue*s, e.g., *NumberValue*, *DateValue*, and *BooleanValue*. At the same time, the concept of *Validator*s was redesigned. In the updated Hybrid Wiki meta-model, attribute definitions have a property *multiplicity* for specifying cardinality constraints, and a *typeConstraint* relation for specifying the value type of corresponding attributes. The set of concrete type constraints was extended in accordance to the newly defined attribute values, i.e., *NumberConstraint*, *DateConstraint*, and *BooleanConstraint*. Furthermore, those type constraints can be refined type-specifically, e.g., the model designer can use regular expressions in *TextConstraints* to ensure that only strings of a particular format can be entered to corresponding attributes.

4.2 UI-related Lessons Learned

In this section, we present the UI-related aspects that should be considered while developing a CIS. The usability of the system is one of the critical quality attributes for such complex systems that involve different stakeholders performing both modeling and data gathering tasks in a collaborative manner. We elaborate the most important UI changes that we introduced in the Hybrid Wiki system in the subsequent subsections.

4.2.1 Displaying Empty Attributes

The Hybrid Wiki approach relies on the feature of attribute suggestion, i.e., when creating new attributes for a wiki page belonging to a specific type tag, the user was offered a list of attribute names. The generation of those suggestions is based on the defined type tag and attribute definitions on the one hand, and on the structure of similar wiki pages on the other. Similarity between wiki pages is defined by the number of type tags they are sharing. Considering again the PMS example in Figure 2, when creating a new publication, the system suggests to add the attributes *title*, *abstract*, and *conference*. However, those suggestions were only visible when the user had the permission to edit the wiki page. If a user does not provide an abstract, the corresponding attribute would not even be displayed to users without write access to the wiki page, giving them the impression that this attribute does not exist.

The Hybrid Wiki projects have shown that this is not a desirable behavior of the system, i.e., the system should show at least all attributes which have a corresponding attribute definition, even if they have no values. In this sense, entities should reflect the user-model as defined by the model designer. Therefore, the updated Hybrid Wiki system not only shows attributes with at least one value in the respective entity's view, but also shows a dummy field for each of the corresponding type's attribute definition, for which the entity does not have any value. Regarding the aforementioned PMS example this would mean that the *abstract* attribute is still shown on the publication view, although it does not have any value.

4.2.2 Optional Deactivation of Free Attributes

As observed in the projects similar to the CPD [12] in which customers had a clear specification and the domain model was first captured and finalized by model designers, the stakeholders felt that allowing free attributes is unnecessary for data owners when gathering application data. In the model-first approach, the user-model captures the desired structures of the data, wherefore the option of free attributes would only distract data owners while capturing data. To ensure a strict top-down approach, we included an option to disable free attributes. Thereby, only attributes which have corresponding attribute definitions can be maintained by data owners. This option is configurable at run time, which allows model designers to control the strictness of the modeling approach, i.e., to switch from a data-first approach to a purely model-first approach seamlessly.

4.2.3 Strict Validation

In the Hybrid Wiki approach, all constraints imposed on the attributes by their corresponding validators including cardinality and data type validators were implemented as *soft constraints*. In other words, users could capture the attribute values for a wiki page defined by the attribute definition of its corresponding type tag definition without necessarily fulfilling their constraints.

A clear indication from our industry partners was to ensure that these cardinality and type constraints were satisfied by data owners while capturing information so as to reduce inconsistencies in the system. The rationale behind this change request was the reduction of distraction for data owners and to enforce them to capture the data as modeled by model designers in the user-model. Since free attributes provide the flexibility for data owners to add new attributes if they are unable to capture the application data using the predefined user-model, we decided to enforce strict validation rules for attributes which have a corresponding attribute definition. For instance, if the data type of the attribute *submissionDeadline* is defined as date, then data owners are only allowed to enter date values. Furthermore, the detailed decisions regarding the data types of attributes could either be determined up-front in a model-first approach or could emerge through contributions from data owners in a bottom-up approach. Moreover, it should also be noted that these strict integrity constraint rules are only applied when the entities are manually created or updated by the users. However, the system still allows to import information from other data sources (e.g., from Microsoft Excel) without being restrained by the aforementioned constraints. The inconsistencies that could arise within the system due to such data import needs to be consolidated by users as further explained in Section 4.2.5.

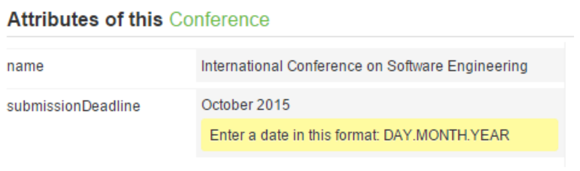
4.2.4 Type-specific Configurable Views

A typical requirement from users of a CIS is the ability to adapt the view of wiki pages to suit their needs. This requirement not only includes changing the logo, color scheme, and fonts but also to be able to (un)hide specific information in wiki pages or to restructure the view of the content in the wiki pages. For instance, stakeholders of the intranet projects as described in Section 3.3 preferred to hide meta information such as last modified date, person responsible for updating an entity, associated tags, and versions of entities, as they felt those details would distract the user. Similarly, users requested to place the list of an entity's attributes (similar to the fact-box of Semantic MediaWiki) at their preferred location, e.g., at the top, bottom, or right side of the page, or to not even show them at all.

The lesson learned through our aforementioned projects regarding the configuration of views is that the respective users want to adapt not only the view of a single wiki page, but the views of all entities of a specific type. For example, the view of an entity representing a publication should be configured only once at type-level. To address this issue in a generic way, we provided features for model designers to be able to configure the view of all entities belonging to a specific type.

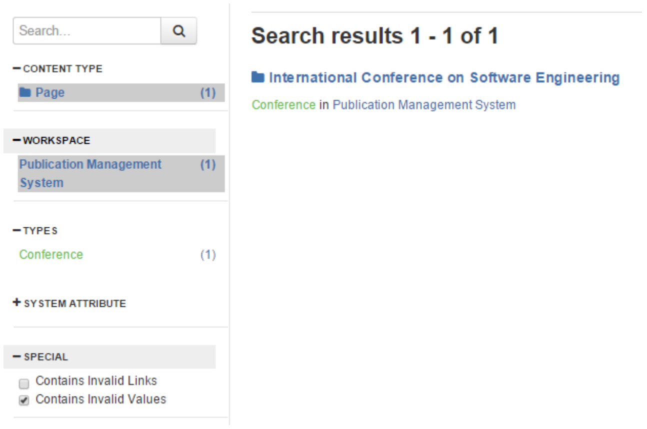
4.2.5 Searchable Inconsistencies

A CIS that supports both the evolution of the user-model and its data will have inconsistencies between the user-model (if already defined) and its associated data at some point of time. For example, when the model designer changes the


Attributes of this Conference

name	International Conference on Software Engineering
submissionDeadline	October 2015 Enter a date in this format: DAY.MONTH.YEAR

Figure 4: Highlighting inconsistencies of an entity's attributes.



Search results 1 - 1 of 1

International Conference on Software Engineering

Conference in Publication Management System

Figure 5: Search facets for supporting the identification of inconsistent entities.

cardinality or an attribute definition's type constraint this could result in inconsistencies in the existing entities of that type. As discussed before, the system also allows inconsistencies when creating an entity during data import (c.f. Section 4.2.3). With this regard, one of the most important lessons learned is that while developing such CIS one should make informed decision not only about when and where to allow inconsistencies but also how to handle these inconsistencies, and how the system should support its users in the consolidation process. As discussed in the earlier version of the Hybrid Wiki approach, validation messages are presented to the users when a wiki page is displayed (c.f., Figure 4). Those validation messages help data owners to ensure that the data in the system is consistent with the corresponding user-model. As these validation messages were only visible when respective wiki pages were visited, it was difficult for users to identify inconsistencies and to handle them appropriately.

To better support the consolidation process, we extended the search functionality of the Hybrid Wiki system with the option to search explicitly for inconsistent entities. Therefore, we added two search facets as depicted in Figure 5 for limiting the set of search results by specific criteria, namely one for identifying invalid values and one for identifying invalid links. In this context, invalid values refer to entities which do not conform to user-model, while invalid links refer to entities which have broken links, i.e., relations to entities which were deleted. Additionally, consolidators can store a search and its parameters, and embed its results in a wiki page. By doing this, users can create and persist their own consolidation views which further facilitates the consolidation process.

5. RELATED WORK

In this section we discuss two types of related work: First, we outline the literature on lessons learned, design principles, or experiences with similar approaches to collaborative information management. Subsequently, we describe further work on adaptions of the Hybrid Wiki meta-model which we do not consider as lessons learned, but as extensions to this concept.

5.1 Design Principles and Studies on Related Approaches

As already pointed out by Matthes et al. [20], there are many semantic wiki approaches related to Hybrid Wikis. All of them share a common set of design principles as defined by Cunningham [7]. For example, one design principle for wikis is the requirement to be organic, which refers to the fact that wiki's data and the underlying model has to be open for editing and evolution. At the same time, the design principle of convergence implies that the evolution of the wiki data and model has to be guided by the system in order to converge to a consistent state. In this sense, those design principles already refer to the alignment of data and model evolution as well as respective tool support. However, they do not describe in detail how to achieve the goal of developing an organic but convergent knowledge management system.

Research about the authoring of wikis investigates how to support the consistent evolution of data and model. For example, Kousetti et al. [15] studied the convergence of ontologies in semantic wikis. They compare existing wiki systems regarding their support for both the data-first and model-first approaches, and propose an approach for fostering the alignment of data and model in the Semantic MediaWiki system. They conclude that respective tool-supported guidance to data and model creation (e.g., suggestions or warnings when editing the data or the model) can foster the convergence of the data and its model. However, their study lacks conceptual and technical details of concrete guidelines which should be considered to achieve better alignment of data and model.

On a related note, Chau and Maurer [6] studied the use a wiki-based experience repository named *MASE* for managing both structured and unstructured information. One of the features of this tool is its capability for supporting self-organization of users in maintaining the data and its corresponding model. However, they do not elaborate on technical measures for improving the support for self-organization of users and alignment of the data with the model.

Apart from research on semantic wikis and corresponding authoring mechanisms, there are also guidelines for collaborative information systems in general. For example, Kraut and Resnick [16] as well as Nielsen [26] propose social design principles for building successful online communities. However, those principles are of an organizational nature (e.g., "Introducing newcomers to a community to members increases interactions", or "Keep few tasks active at any given time") and they are not concrete technical guidelines on how to build such a system.

5.2 Related Adaptions of Hybrid Wikis

Apart from the lessons learned as described in Section 4, recent research projects revealed the opportunities for two extensive extensions to the Hybrid Wiki approach.

The first extension of the Hybrid Wiki approach is the implementation of an expression language for defining queries and rules based on the user-defined data model [24]. The requirement for such an expression language originated from the domain of EAM: EA stakeholders wanted to be able to define EA metrics [13] based on the EA data they were maintaining and designing in the Hybrid Wiki system. Consequently, Monahov et al. [24] and Reschenhofer et al. [31] designed a language fulfilling the respective requirements. Thereby, they extended the Hybrid Wiki meta-model by concepts for integrating the designed expression language, e.g., *DerivedProperty* to define attributes whose values are automatically computed by the system, or *CustomFunction* to define reusable and parameterizable functions implemented by the expression language. By using this language, enterprise architects are also able to analyze the temporal evolution of the defined EA metrics [4].

The second research initiative based on the Hybrid Wiki approach is a data-centric approach by Hauder et al. [10] to support knowledge-intensive processes [8]. Again, this extension to the Hybrid Wiki meta-model was motivated from applications in EAM [11], and introduces the concept of *Task*s for guiding users in maintaining the Hybrid Wiki's data. Furthermore, Hauder et al. define *TaskDefinition*s representing reusable work plans. Those work plans can be refined iteratively, and define a light-weight process structure consisting of tasks potentially having dependencies between each other. In this sense, tasks and work plans are the process analogies to the entity and type concepts of the Hybrid Wiki meta-model in Figure 3.

6. CONCLUSION

This work presents the lessons learned from five years of applying the collaborative Hybrid Wiki approach in multiple industrial and research projects in the domains of Enterprise Architecture Management, Collaborative Product Development, and Collaborative Content Management. We understand lessons as challenges and issues we faced in those projects regarding the co-evolution of data and model, and how those were addressed by concrete conceptual and technical adaptions of the applied Hybrid Wiki approach. While we relate the lessons learned to applications of the Hybrid Wiki meta-model as well as to concrete adaptions of the same, we believe that they also hold for similar approaches to systems which combine data- and model-first approaches in a collaborative manner. In this sense, they constitute the answers to both research questions raised in Section 1.

There are three main conclusions which we draw from our experiences from the applications of the Hybrid Wiki approach. First, for software systems supporting a collaborative approach to model and data evolution, finding the right balance between data- and model-first approaches to modeling is decisive. The practical applications of the Hybrid Wiki approach revealed that in early stages of the user-model design, a focus on the data-first approach enables model designers to harness collective intelligence among the system's users and to utilize each individual's domain-specific knowledge. However, as soon as the user-model reaches a certain degree of maturity, the design space should be restricted in order to enforce a convergence of the user-model, implying a shift to a stricter model-first approach.

Second, the co-evolution of both the user-model and its data yields to inconsistencies between them. One important success factor of software systems supporting this co-evolution is the integration of adaquate data consolidation tools and techniques. For example, such a software system should support users in identifying inconsistent data, aligning it to the user-model, and—whenever possible and reasonable—in automating certain consolidation steps.

Third, as revealed by the concept-related lessons learned, a conceptual model enabling the co-evolution of user-model and data has to have the right balance between simplicity and expressiveness. For example, based on the feedback from users of the system, we had to reduce the complexity of the initial Hybrid Wiki meta-model by simplifying the entity-type relationship. However, at the same time we added expressiveness by extending the meta-model with additional attribute types. This indicates that the basic concepts of a pragmatic approach to model and data co-evolution should be usable and understandable by a broad spectrum of both data owners and model designers. In the context of the Hybrid Wiki approach, this refers to the concepts *Entity*, *Attribute*, *Type*, and *Attribute Definition*. More elaborated features and concepts improving the expressiveness of such an approach should be reserved for experienced users, and hidden from the others, e.g., type-specific constraints as described in Section 4.1.5.

In addition to the presented lessons learned, the application of the Hybrid Wiki approach revealed that in many cases users would like to have stakeholder-specific views and case-specific logic based on the foundations provided by the Hybrid Wiki approach. Therefore, one focus of our future research activities is the endeavor to design a platform implementing the Hybrid Wiki system and making its services reusable and integratable into other software systems.

7. REFERENCES

[1] F. Ahlemann, E. Stettiner, M. Messerschmidt, and C. Legner. *Strategic Enterprise Architecture Management.* Springer-Verlag, 2012.

[2] C. Atkinson and T. Kühne. Model-Driven Development: A Metamodeling Foundation. *IEEE Software*, 20(5):36–41, 2003.

[3] C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[4] M. Bhat, T. Reschenhofer, and F. Matthes. Tool Support for Analyzing the Evolution of Enterprise Architecture Metrics. *Proceedings of the International Conference on Enterprise Information Systems*, 2015.

[5] S. Buckl, F. Matthes, C. Neubert, and C. M. Schweda. A Wiki-based Approach to Enterprise Architecture Documentation and Analysis. *Proceedings of the European Conference on Information Systems*, 2009.

[6] T. Chau and F. Maurer. A Case Study of Wiki-based Experience Repository at a Medium-sized Software Company. *Proceedings of the International Conference on Knowledge Capture*, pages 185–186, 2005.

[7] W. Cunningham. Wiki Design Principles, 2011.

[8] T. H. Davenport. *Thinking for a Living: How to Get Better Performances and Results from Knowledge Workers.* Harvard Business Press, 2013.

[9] X. L. Dong and D. Srivastava. Big Data Integration. *Proceedings of the International Conference on Data*

Engineering, pages 1245–1248, 2013.

[10] M. Hauder, R. Kazman, and F. Matthes. Empowering End-Users to Collaboratively Structure Processes for Knowledge Work. *Proceedings of the International Conference on Business Information Systems*, 2015.

[11] M. Hauder, D. Münch, F. Michel, A. Utz, and F. Matthes. Examining Adaptive Case Management to Support Processes for Enterprise Architecture Management. *Proceedings of the Enterprise Distributed Object Computing Conference Workshops and Demonstrations*, pages 23–32, 2014.

[12] M. Hauder, S. Roth, F. Matthes, A. Lau, and H. Matheis. Supporting Collaborative Product Development Through Automated Interpretation of Artifacts. *Proceedings of the International Symposium on Business Modeling and Software Design*, 2013.

[13] M. Hauder, S. Roth, C. Schulz, and F. Matthes. Current Tool Support for Metrics in Enterprise Architecture Management. *Proceedings of the DASMA Software Metrik Kongress*, 2013.

[14] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design Science in Information Systems Research. *Management Information Systems Quarterly*, 28(1):75–105, 2004.

[15] C. Kousetti, D. E. Millard, and Y. Howard. A Study of Ontology Convergence in a Semantic Wiki. *Proceedings of the International Symposium on Wikis and Open Collaboration*, pages 17:1–17:10, 2008.

[16] R. E. Kraut, P. Resnick, S. Kiesler, M. Burke, Y. Chen, N. Kittur, J. Konstan, Y. Ren, and J. Riedl. *Building Successful Online Communities: Evidence-based Social Design*. MIT Press, 2012.

[17] H. Matheis. SmartNet Navigator and Application Guidelines. *Seventh Framework Programme*, 2013.

[18] F. Matthes and C. Neubert. Enabling Knowledge Workers to Collaboratively Add Structure to Enterprise Wikis. *Proceedings of the European Conference on Knowledge Management*, 2011.

[19] F. Matthes and C. Neubert. Wiki4EAM - Using Hybrid Wikis for Enterprise Architecture Management. *Proceedings of the International Symposium on Wikis and Open Collaboration*, 2011.

[20] F. Matthes, C. Neubert, and A. Steinhoff. Hybrid Wikis: Empowering Users to Collaboratively Structure Information. *Proceedings of the International Conference on Software and Data Technologies*, 2011.

[21] A. P. McAfee. Enterprise 2.0: The Dawn of Emergent Collaboration. *MIT Sloan Management Review*, 47(3):21–28, 2006.

[22] S. McGinnes and E. Kapros. Conceptual Independence: A Design Principle for the Construction of Adaptive Information Systems. *Information Systems*, 47:33–50, 2015.

[23] T. D. Meijler, J. P. Nytun, A. Prinz, and H. Wortmann. Supporting Fine-grained Generative Model-driven Evolution. *Software & Systems Modeling*, 9(3):403–424, 2010.

[24] I. Monahov, T. Reschenhofer, and F. Matthes. Design and Prototypical Implementation of a Language Empowering Business Users to Define Key Performance Indicators for Enterprise Architecture Management. *Proceedings of the Trends in Enterprise Architecture Research Workshop*, 2013.

[25] A. I. Mørch, G. Stevens, M. Won, M. Klann, Y. Dittrich, and V. Wulf. Component-based Technologies for End-user Development. *Communications of the ACM*, 47(9):59–62, 2004.

[26] M. Nielsen. *Reinventing Discovery: The New Era of Networked Science*. Princeton University Press, 2012.

[27] M. Q. Patton. Evaluation, Knowledge Management, Best Practices, and High Quality Lessons Learned. *American Journal of Evaluation*, 22(3):329–336, 2001.

[28] E. Rahm and P. A. Bernstein. An Online Bibliography on Schema Evolution. *ACM SIGMOD Record*, 35(4):30–31, 2006.

[29] S. Rehm, T. Reschenhofer, and K. Shumaiev. IS Design Principles for Empowering Domain Experts in Innovation: Findings From Three Case Studies. *Proceedings of the International Conference on Information Systems*, 2014.

[30] T. Reschenhofer, I. Monahov, and F. Matthes. Application of a Domain-Specific Language to Support the User-Oriented Definition of Visualizations in the Context of Collaborative Product Development. *Proceedings of the International Conference on Interoperability for Enterprises Systems and Applications*, 2014, 2014.

[31] T. Reschenhofer, I. Monahov, and F. Matthes. Type-Safety in EA Model Analysis. *Proceedings of the Trends in Enterprise Architecture Research Workshop*, 2014.

[32] J. F. Roddick, L. Al-Jadir, L. Bertossi, M. Dumas, H. Gregersen, K. Hornsby, J. Lufter, F. Mandreoli, T. Männistö, and E. Mayol. Evolution and Change in Data Management - Issues and Directions. *ACM SIGMOD Record*, 29(1):21–25, 2000.

[33] S. Roth, M. Hauder, and F. Matthes. Collaborative Evolution of Enterprise Architecture Models at Runtime. *Proceedings of the Workshop on Models at Runtime*, 2013.

[34] M. Spahn, C. Dörner, and V. Wulf. End User Development: Approaches Towards a Flexible Software Design. *Proceedings of the European Conference on Information Systems*, pages 303–314, 2008.

[35] M. van Oosterhout, E. Waarts, and J. van Hillegersberg. Change Factors Requiring Agility and Implications for IT. *European Journal of Information Systems*, 15(2):132–145, 2006.

[36] P. Vitharana. Risks and Challenges of Component-based Software Development. *Communications of the ACM*, 46(8):67–72, 2003.

[37] V. Wulf and M. Rohde. Towards an Integrated Organization and Technology Development. *Proceedings of the Conference on Designing Interactive Systems*, 1995.