Empowering End-users to Collaboratively Manage and Analyze Evolving Data Models

Full Paper

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

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

Abstract

In order to empower end-users to make well-founded decisions based on domain-specific knowledge, companies use end-user oriented business intelligence (BI) software like spreadsheets. Moreover, many decisions require the collaboration of multiple and autonomous knowledge workers. However, prevalent BI software does not provide elevated collaboration features as known from traditional Web 2.0 technologies. There is also a lack of research on how to integrate collaboration features into BI systems, and which challenges arise as a consequence.

In the paper at hand we address this issue by proposing the Spreadsheet 2.0 approach, which integrates Web 2.0 features with the spreadsheet paradigm as most-common representative of end-user-oriented business intelligence tools. Therefore, we derive requirements for a Web 2.0-based approach to collaborative BI, and present the conceptual design for a Spreadsheet 2.0 solution. Subsequently, we demonstrate a corresponding prototypical implementation, and elaborate on key findings and main challenges identified by its application and evaluation.

Keywords

Spreadsheets, End-User Development, Web 2.0, Evolving Data Model, Decision Support.

Introduction

Due to the digital transformation and consequentially an increasing amount of digital information, the importance of knowledge workers which are able to perform complex information management and analysis tasks autonomously rises considerably (Davenport 2013). These knowledge workers have to adapt their work environment continuously to changing business requirements, technology innovations, and legal regulations. In this sense, end-user development (Lieberman et al. 2006) allows knowledge workers to adapt their software tools by themselves. While end-user development imposes significant challenges for knowledge workers (Pinnington et al. 2007), it also improves their autonomy and potentially their efficiency (Mørch et al. 2004).

The most prominent class of end-user developed information systems are spreadsheets (Pemberton and Robson 2000). They enable knowledge workers to mine domain-specific data by analyzing and visualizing it in an end-user-friendly way. Related studies revealed that nearly all enterprises are using spreadsheets to support various business processes (Bradley and McDaid 2009), e.g., financial reporting. In contrast to traditional business intelligence (BI) solutions which provide standard reports predefined by IT experts, spreadsheets are definable and adaptable by end-users. This allows them to define tailored data analysis operations and data views. At the same time spreadsheets suffer from several shortcomings when adopted as business applications (Reschenhofer and Matthes 2015), e.g., their unmanaged evolution, or lacking collaboration support. Especially the latter one is critical considering that well-found decisions should be based on the collective intelligence of multiple knowledge workers with diverse business expertises. In this context, collaborative decision making means not only that the underlying data is managed collaboratively, but also that its analysis and visualization is performed in a collaborative way, e.g., by sharing reusable analysis functions (cf. formulas in spreadsheets) or visualizations. However, prevalent spreadsheet solutions and BI solutions in general do not provide such sophisticated collaboration features (Kaufmann and Chamoni 2014).

In the paper at hand, we propose a Web 2.0-based approach to collaborative and end-user-oriented business intelligence. In the style of *Enterprise 2.0 (McAfee 2009)* which describes the adoption of collaborative technology in the corporate context, and based on spreadsheets as the most common representatives of end-user-oriented BI, we name our approach *Spreadsheet 2.0*. Thereby, we present the conceptual design and prototypical implementation of a Spreadsheet 2.0 application whose aim is to integrate the collaborative nature of Web 2.0 technology and the spreadsheet paradigm. In this context, the spreadsheet paradigm refers to semi-structured and flexible data modelling as well as the end-user-driven and responsive definition of analysis functions and data visualizations. We also present results of the evaluation of the Spreadsheet 2.0 approach and the corresponding prototype.

The underlying research objective is to elaborate on how collaborative BI can be implemented on the foundations of prevalent Web 2.0 technology (Hertogh et al. 2011) and by the adoption of the spreadsheet paradigm, and which challenges arise as a consequence of integrating those two worlds. We organized our work in accordance to the Design Science Research methodology by Hevner et al. (2004). Therefore, we provide an overview over related work in the field of collaborative BI and relate our approach to existing ones. Subsequently, we describe the rationale behind coming up with a Spreadsheet 2.0 approach as well as its conceptual design. We demonstrate the prototypical implementation by a practical example. Afterwards, we outline the prototype's application in the domains of Enterprise Architecture Management (EAM) and Collaborative Product Development (CPD), and summarize the key findings from interviews with knowledge workers from three companies. Finally, we conclude our work by outlining the key features of the proposed solution and by mapping them to common Web 2.0 requirements.

Related Work

The topic of collaborative BI and related approaches has already been subject to research for years. Mertens and Krahn (2012) studied shortcomings of analytical information systems (AIS), which they describe as BI tools allowing technology-savvy users to analyze huge integrated datasets, and derive requirements for making them usable for business users (for the remainder of the paper, the terms *business user, end-user*, and *knowledge worker* are used synonymously). Based on those requirements, they propose a conceptual system architecture for an end-user-oriented AIS which integrates different data sources, ontologies, and analysis components. While Mertens and Krahn (2012) aim for making traditional BI functionality accessible and applicable for business users, the paper at hand focuses on fostering the collaboration aspect of end-user-oriented BI. Nevertheless, they already outline requirements for their AIS which are also important for the Spreadsheet 2.0 approach, e.g., a flexible metadata model, domain-specific instances of the same, and support for ad-hoc analysis.

On a related note, Spahn et al. (2008) argue that traditional BI systems are too complex to be adapted by end-users to a changing environment and to their individual and domain-specific needs. To address this issue, they propose a system architecture based on ontologies. In this way, their system implements a semantic layer abstracting data from heterogeneous data sources and providing it as an integrated and comprehensible data model which is understandable by end-users. On the top of this semantic layer, they present a visual query designer which empowers end-users to define queries in an ad-hoc manner. Similar to Mertens and Krahn (2012), Spahn et al. (2008) utilize the concept of ontologies in order to abstract the technical representation of data and to make it understandable and usable by end-users. Then again, their tool is targeting single end-users rather than a group of knowledge workers collaborating in the definition of an analysis model in general or queries in particular.

A very similar ontology-based approach is implemented by Sell et al. (2005), who propose a web-based architecture for an AIS. In doing so, they address the lack of extensibility of related tools with respect to their exploratory capabilities, or the representation of information. Their layered architecture builds on different ontologies which can be transformed by extensible functional modules, which in turn are exposed via web services and thus reusable by multiple client applications. However, Mertens and Krahn (2012), Spahn et al. (2008), and Sell et al. (2005) do not discuss how those ontologies and functional modules could be managed in a collaborative environment, or which consequences arise from the adoption of this service-oriented architecture in a collaborative environment at all.

Actually, as revealed by Kaufmann and Chamoni (2014) in their literature study about collaborative BI, those collaboration-related challenges are addressed only by very few publications,: Dayal et al. (2008)

outline requirements for a collaborative BI platform, e.g., the need for the integration of ontologies and rich metadata, or the requirement of providing tools for querying and analyzing historical and real-time information. To enable collaboration among knowledge workers, they propose and implement a virtual three-dimensional room in which cooperating knowledge workers communicate via speech or gesture.

On another note, Rizzi (2012) studies BI networks involving multiple companies which collaborate by connecting their individual and autonomous BI systems. Those BI systems can share different kinds of information which potentially leads to new inter-organizational relationships. Therefore, Rizzi (2012) studies the collaboration of companies or organizations in common BI initiatives rather than the collaboration of multiple knowledge workers in their interrelated BI activities as intended by this paper.

Berthold et al. (2010) also address the latter – intra-organizational – view on collaborative BI: They propose an architecture for business intelligence, which in contrast to Spahn et al. (2008), Mertens and Krahn (2012), and Sell et al. (2005) focuses on end-user-oriented BI in a collaborative environment. Based on an abstract data integration layer providing flexible and enrichable data structures, this architecture defines an ad-hoc analysis and collaboration environment empowering end-users to collaboratively make decisions. However, they provide only a very high-level view of a collaborative BI system. In contrast, in the paper at hand we propose a conceptual model and prototype for a collaborative BI tool, and thus a more concrete perspective.

Design and Implementation of Spreadsheets 2.0

In this section, we describe the design rationale of the Spreadsheet 2.0 approach, elaborate on its conceptual design, and present selected aspects of the prototypical Spreadsheet 2.0 implementation.

Design Rationale

Kaufmann and Chamoni (2014) claim that there is a lack of research on how to integrate collaboration features into BI systems, and how collaboration affects the different components of BI systems. In order to address this issue, we base the design the Spreadsheet 2.0 tool on a three-layered architecture consisting of a data layer, analysis layer, and visualization layer. This conceptual architecture is line with related work about end-user-oriented BI architectures as discussed in the previous section. Based on those layers, we adopt the six defining structural capabilities of Web 2.0 technology as derived by Hertogh et al. (2011) to collaborative BI to come up with basic requirements for a Spreadsheet 2.0 application:

1) **Reusability of data, functionality, and visualizations**: In a Spreadsheet 2.0 application, the artifacts of all layers should be reusable and combinable by oneself and by other users.
2) **Adaptability of data, functionality, and visualizations**: End-users should be able to adapt the artifacts of each architecture layer at any time, provided they have respective access rights.
3) **Collaborative content creation and modification**: A Spreadsheet 2.0 application facilitates the collaborative creation and management of data, analysis functions, and visualizations.
4) **No predefined structure of content**: A collaborative Spreadsheet 2.0 application should not impose a predefined structure on its content, i.e., it should support knowledge workers to dynamically enrich content with additional, not yet defined structure.
5) **Responsive and personalized user interface (UI)**: In this context, responsiveness of an UI refers to immediate feedback to users when performing certain actions, while a personalized UI takes into account the user's access rights, personal settings, etc.
6) **Gathering of collective intelligence**: A Spreadsheet 2.0 application should involve knowledge workers with different domain knowledge in content creation and management.

Conceptual Design

In this section, we present the conceptual meta-model of a Spreadsheet 2.0 including data-related, analysis-related, and visualization-related concepts (c.f. Figure 1 and Figure 4). The presented conceptual model is minimal in the sense that it outlines the main parts of the Spreadsheet 2.0 architecture, but it does not show concepts which are not relevant for this work's discussion. We design the meta-model

based on our experience in more than 10 years of applying Web 2.0 technology in corporate environments for enabling collaborative content and model management.

The conceptual design of a Spreadsheet 2.0 is based on the adaption of a classical Web 2.0 concept, namely Hybrid Wikis by Matthes et al. (2011). In Hybrid Wikis, initially unstructured wiki pages are collaboratively and iteratively enriched by structure in the form of type annotations, attributes, and constraints. All changes are captured, tracked, and reversible through an explorable version history mechanism. Conceptually, the Hybrid Wiki meta-model (c.f. grey-colored concepts in Figure 1) consists of *workspaces* (representing wikis) and *entities* (representing wiki pages), which have *attributes* of different data types. The concepts *entity type* and *attribute definition* define the schema and constraints for the data objects. Users can dynamically create entity types and assign them to entities in order to impose a certain structure, i.e., the entities should have attributes which correspond to the attribute definitions of the assigned entity type. The attribute definitions can define data type constraints, i.e., they force corresponding attributes to be of the defined type. Figure 1 shows that a data type is either a basic type (e.g., number, date, or string), a composite type (e.g., a collection of strings, or a complex nested data structure), or a user-defined entity type. Therefore, knowledge workers can collaboratively manage workspaces which represent domain-specific data models including classes, attributes, and relationships. In this way, the Hybrid Wiki concepts forms the data layer of the Spreadsheet 2.0 application.

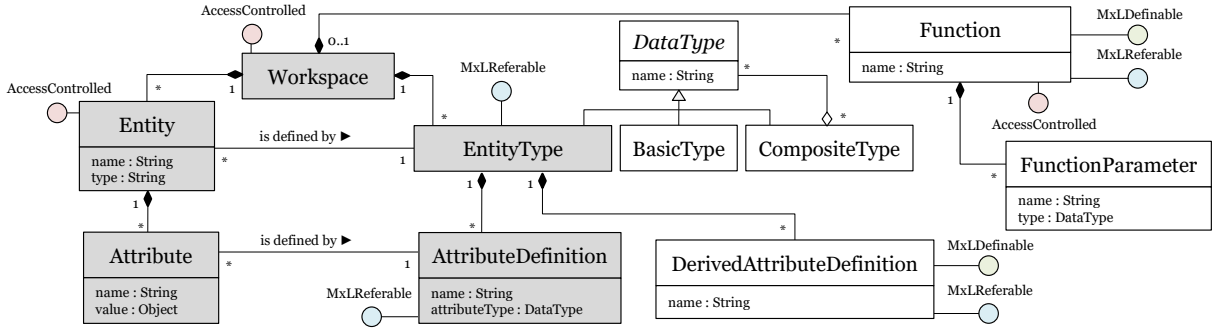

Figure 1. The Spreadsheet 2.0 data and analysis meta-model based on Hybrid Wikis (grey-colored) by Matthes et al. (2011). The interfaces are defined in Figure 2 and Figure 3.

As shown in Figure 1, we extend the Hybrid Wiki meta-model by the concepts *DerivedAttributeDefinition* and *Function* (which can have an arbitrary number of parameters), which form the analysis layer of the Spreadsheet 2.0 architecture. The main reason for adding the derived attribute definition concept is its seamless integration into the Hybrid Wiki approach and the corresponding prototype, while the function concept was added in order to address the reusability requirement as described in the previous section. Derived attribute definitions define attributes whose values are not manually maintained by end-users, but automatically computed based on an end-user-defined business rule. Functions represent parameterized operations for encapsulating reusable functionality. They can either be assigned to a workspace (if they represent domain-specific functionality), or they can be defined globally (if the functionality should be shared across workspaces and thus business domains). Derived attribute definitions and functions are definable by end-users, which implement them by a functional and model-based expression language (MxL) as defined by Reschenhofer et al. (2014). MxL is based on the Hybrid Wiki concepts, and empowers end-users to define expressions at runtime based on the domain-specific data model and by using arithmetic, statistical, and query operations. By using this language, users can also analyze the temporal evolution of the data and its model by accessing the data's version history (Bhat et al. 2015).

MxL expressions refer to the data model-related elements of the meta-model in Figure 1, i.e., entity types and attribute definitions, but also to other functions and derived attribute definitions. For example, when modelling a *Person* as an entity type with an attribute definition *Birth date*, one can define a derived attribute definition *Age* based on the attribute definition *Birth date* and the current date. In this way, MxL expressions define semantic dependencies between elements specified by such an expression (*MxLDefinable*), and the elements to which an expression refers to (*MxLReferable*). In the aforementioned example, there is a semantic dependency from *Age* to *Birth date*. These semantic dependencies are explicitly captured in Figure 2 showing two interfaces *MxLDefinable* and

MxLReferable. The relation from the former to the latter is automatically derived from the *MxLDefinable*'s expression. This is enabled by MxL's static type-safety, i.e., when a *MxLDefinable* is created, the Spreadsheet 2.0 application validates the static semantics of the corresponding expression and resolves all references. Figure 1 shows that functions and derived attribute definitions are *MxLDefinables*, while they – in addition to entity types and attribute definitions – are also implemented as *MxLReferables*. A main purpose of the analysis of *MxLDefinables* and the derivation of semantic dependencies is the automated refactoring of MxL expressions on changes of corresponding *MxLReferables*. For example, if the attribute definition *Birth date* would be renamed to *Birthday*, the derived attribute definition *Age* would become invalid. Since the system knows the semantic dependencies between them, it can automatically adapt the MxL expression of affected *MxLDefinables* to changes of corresponding *MxLReferables*. Particularly in the context of Hybrid Wikis and evolving data models in general, this feature is highly relevant to keep consistency of the analysis model.

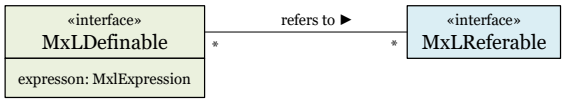

Figure 2. Definition of interfaces *MxLDefinable* and *MxLReferables*.

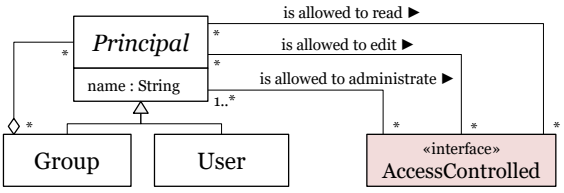

Figure 3. A general authorization model as the access control concept for Spreadsheets 2.0

All elements captured by the Spreadsheet 2.0 meta-model in Figure 1 are defined and managed collaboratively by multiple knowledge workers. In order to coordinate and control which knowledge workers can access which elements, the conceptual Spreadsheet 2.0 meta-model includes concepts for discretionary access control. As shown in Figure 3, the authorization model defines an interface *AccessControlled* representing all elements within the Spreadsheet 2.0 application whose accessibility can be defined explicitly. Principals – which are either users or groups – can be assigned to access-controlled elements as readers (have read-access), editors (can edit the element), or administrators (can change the access rights of the element). Users having neither of those roles for an access-controlled element cannot see it. The access rights also apply to the execution of MxL expressions, i.e., users can only query entities to which they have at least read-access. This leads to the fact that the execution of MxL expressions might lead to different results for different users, which enables the definition of user-specific and personalized views and visualizations.

As defined in Figure 1, the only elements whose accessibility is explicitly controlled and thus implementing the *AccessControlled* interface are workspaces and entities. The access rights for other elements are implicitly derived from respective container elements, i.e., attributes derive their access rights from the owner entity, while other elements derive them from the corresponding workspace. For example, if a user is a reader of a workspace, he is also reader of the corresponding data model consisting of entity types, attribute definitions, and derived attribute definitions. Readers of MxLDefinables (functions and derived attribute definitions) are also allowed to execute them, while only editors of them are allowed to change the corresponding MxL expression.

On the top of the data and analysis layer of the Spreadsheet 2.0 architecture (c.f. Figure 1), we define the visualization layer as shown in Figure 4. Those concepts are inspired by the abstract view model as defined by Hauder et al. (2012). In the context of Spreadsheets 2.0, end-user-developers with respective web application engineering skills can define generic and reusable *visualization types* by using common visualization libraries and frameworks for the web. In order to deploy visualization types to the Spreadsheet 2.0 application and to make them instantiable for business users, they have to specify *data binding definition*s describing the interface for the visualization type, i.e., which kind of data is visualized. For example, a visualization type *bar chart* could define a data binding of type *Sequence<Number>* (i.e., a collection of numerical values), whereas each number represents the size of one bar.

To create domain-specific *visualizations*, users have to choose from the set of available visualization types, and instantiate them by providing a *data binding* for each of the visualization type's data binding definition. For this purpose, they have to use MxL to define queries based on the underlying domain-specific data model. Therefore, the data binding concept implements the *MxLDefinable* interface, and

thus has semantic dependencies to the concepts in Figure 1. When instantiating visualizations, the users have to make sure that the output of MxL queries for each data binding complies to the *parameterType* of the respective data binding definition. For example, if the data binding definition is of type *Sequence<Number>*, the respective MxL query has to return a collection of numerical values.

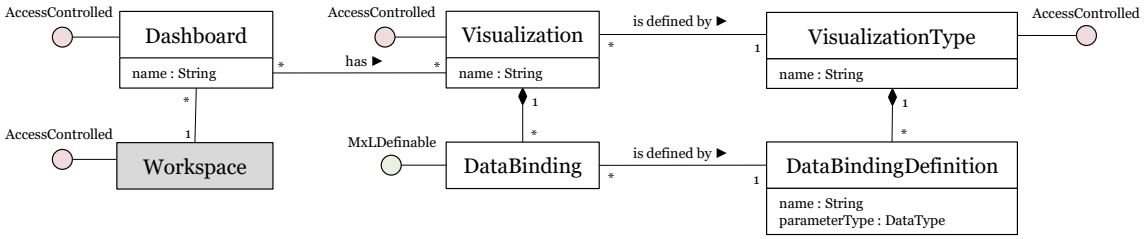

Figure 4. The Spreadsheet 2.0 visualization meta-model based on the concepts in Figure 1

In the Spreadsheet 2.0 application, multiple visualizations are composed to dashboards. Visualizations can even be contained in more than one dashboard, which allows the reusability of the visualization's (visual) configuration and data bindings. Dashboards are considered to be domain-specific, wherefore they are assigned to a specific workspace. The dashboard, visualization, and visualization type concepts implement the *AccessControlled* interface as defined in Figure 3, i.e., users can explicitly set readers, editors, and administrators for those elements. Apart from this, the MxL queries which bind the data to the visualizations only have access to entities for which the user has at least the reader role.

Illustration of the Spreadsheet 2.0 Prototype

In this section, we present selected aspects of the Spreadsheet 2.0 prototype by illustrating its application in an exemplary scenario in the domain of Enterprise Architecture Management (EAM). Since the original Hybrid Wiki approach was already successfully applied as an EAM tool (Matthes and Neubert 2011), the paper at hand focuses on aspects related to the analysis and visualization layer.

In EAM, different stakeholders (e.g., enterprise architects, data owners, business users) collaboratively document the enterprise architecture of their organization. In our example, we assume that the collaborative and iterative design approach by using the Hybrid Wikis concepts yielded to the data model in Figure 5, which is inspired by the work from Schneider et al. (2015). In this example, the data model captures all business applications which are used within an organization. Corresponding data owners are maintaining their meta data, e.g., function points indicating the functional scope of a business application. Furthermore, the business application is related to the databases it is using, while information flows represent data which is transferred from one business application to another one.

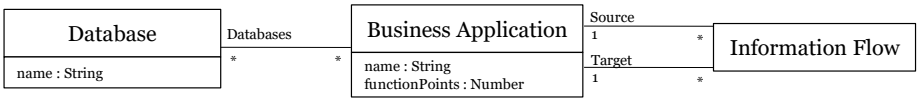

Figure 5. An exemplary data model for documenting application landscapes in EAM

Schneider et al. (2015) describe several EA metrics which can be defined based on the data model in Figure 5. For example, an enterprise architect named Adam has the goal to standardize the application landscape (AL) with respect to data bases. Therefore, he would like to define the database complexity metric based on the *Shannon entropy*. Since he knows that a colleague is interested in the standardization of the AL with respect to operating systems (which are not yet captured by the data model), he defines the *Shannon entropy* as a generic reusable function, whose only parameter is a list of any kind of objects. Based on the data model as illustrated in Figure 1, Adam creates a dashboard, and chooses a visualization type for showing the database complexity metric. In this example, he chooses a very simple one which is just displaying a number, and which only has one numerical data binding to be initiated. Figure 6 shows how Adam configures the visualization to show the database complexity metric (Schneider et al. 2015). Thereby, the system supports the user in defining the MxL expression by providing an auto completion feature, which not only proposes available operations, but also data model-related elements (e.g., entity types like *Business Application*, or attribute definitions like *Databases*) including proper description.

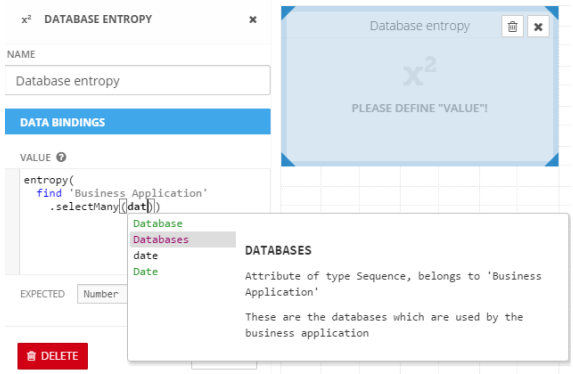

Figure 6. Instantiation of a visualization for database complexity metric

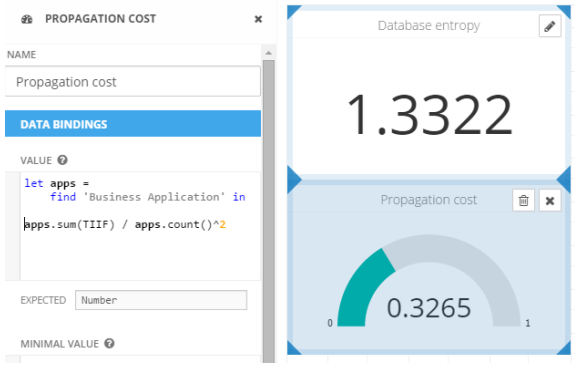

Figure 7. Instantiation of a visualization for the AL propagation costs

In parallel to Adam's activities, another enterprise architect named Bill is interested in the structural complexity of the organization's AL which is defined by business applications and their information flows. Bill wants to determine the virtual propagation costs of business applications, which is defined by the proportion of business applications in the AL, which are affected by changes to a random one, whereas changes are propagated via information flows (Schneider et al. 2015). For this purpose, Bill defines derived attributes for the entity type *Business Application* which compute the number of transitive incoming and outgoing information flows, which he names TIIF and TOIF respectively. Subsequently, he can extend Adam's dashboard by a new visualization. As visualization type, he chooses a Gauge chart and instantiates the data binding definitions as illustrated in Figure 7. This figure also illustrates that the visualization is already generated during the configuration, which ensures that the user gets immediate visual feedback when defining the complexity metric with MxL.

Some days later, Charlie – another enterprise architect in the same organization – found out that there is an AL complexity metric inspired by the Design Structure Matrix (DSM) which classifies the applications into *control*, *core*, *shared*, and *periphery* components (Schneider et al. 2015). Again, this metric is based on transitive incoming and outgoing information flows of business applications. Therefore, he just reuses the derived attribute definitions as already defined by Bill, and creates another (pie chart) visualization in Adam's dashboard, whose final state is depicted in Figure 8.

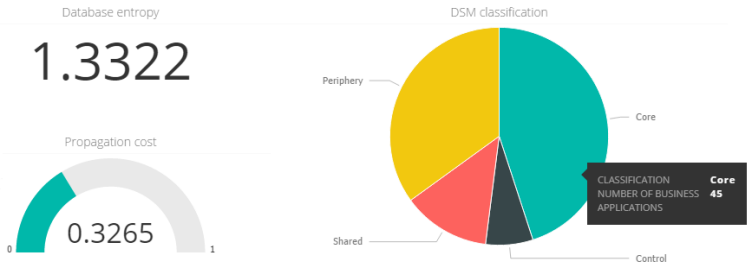

Figure 8. A dashboard collaboratively designed by multiple enterprise architects

The example of Adam, Bill, and Charlie shows how knowledge workers can share functionality and visualizations among each other, while data owners and other stakeholders are collaboratively maintaining the underlying data model. The Spreadsheet 2.0 prototype ensures that the data, analysis, and visualization models remain consistent, although parts of them are continuously evolving over time.

Practical Experiences and Evaluation

The prototype was already applied in several research projects. For example, Schneider et al. (2015) used the prototype for implementing the different complexity metrics in the domain of EAM (the example in the previous section was inspired by this use-case). Thereby, multiple researchers iteratively developed EA metrics and applied them to real data provided by four companies. During the design process of the

metrics, the underlying data model had to be adapted multiple times due to company-specific characteristics of the provided data sets. The automated refactoring of metrics ensured their consistency.

On another note, the predecessor of the presented Spreadsheet 2.0 prototype was applied in an EU project to support networks of enterprises in collaborative product development (CPD) projects (Rehm et al. 2014). In this version of the prototype, the visualization layer was not yet implemented as it is described in this work. Nevertheless, end-users were still able to define web-based visualizations by simply generating HTML-markup based on the outcome of MxL queries. In this way, users of the systems defined interactive views for the coordination of their projects, whereas these views visualize the status of development project based on the status of related tasks and meetings. However, the difficult maintainability and reusability of this kind of visualizations was one of the main drivers for the development of the visualization layer as described in this work.

Apart from concrete applications of the Spreadsheet 2.0 prototype, we interviewed knowledge workers in three different German companies, namely an enterprise architect of an IT services provider (5,000 - 10,000 employees), an IT infrastructure manager of an investment company (10,000+ employees), and data quality manager of a logistics company (10,001+ employees). We demonstrated the prototype by the implementation of domain-specific example dashboards, e.g., an EAM dashboard for the enterprise architect. We asked them for their assessment on the analysis and visualization layer of the prototype.

With respect to the analysis and visualization layer of the Spreadsheet 2.0 prototype, the key findings of the interviews as well as main issues identified in the prototype's aforementioned applications are:

- The intention when designing the authorization model (c.f. Figure 3) was to ensure the usability by end-users. Therefore, we kept it rather simple by only control the access to workspaces and entities explicitly, while the access control rules for other objects are implicitly determined. However, certain cases required a more fine-grained authorization model, e.g., to control the access to single attributes or functions.
- When performing queries via MxL expressions, users only have access to entities where they have at least read-access. Consequently, different users might get different results when performing the same query. The same is true for visualizations, since they are bound to the system's data via MxL queries. While this is a desirable behavior for defining personalized views and visualizations, some use-cases require the definition of queries and views which produce the same result for each user, regardless of the user's access rights. A solution approach would be to allow the specification of evaluation identities for *MxLDefinables*: If an MxL expression is executed, the specified identity is impersonated instead of evaluating the expression as the current user.
- MxL is a functional and object-oriented language inspired by UML's Object Constraint Language (OCL) and Microsoft's Language Integrated Queries (LINQ). Although end-users in many domains are experienced in using the formula language of spreadsheet applications, MxL seems to be too technical and complex in some cases. Therefore, the Spreadsheet 2.0 prototype should provide user-friendlier tools for defining MxL expressions, e.g., to bind data to visualizations. However, in domains where users are more tech-savvy (e.g., EAM), those users even prefer a language like MxL which is more powerful than spreadsheet formulas. Therefore, the system should provide both an expert view and a user-friendly view to satisfy both types of users.

Conclusion

This paper presents our work on a concept for a Spreadsheet 2.0 application, which integrates the spreadsheet paradigm with Web 2.0 features in order to empower end-users to collaboratively manage, analyze, and visualize domain-specific data. The paper outlines the conceptual design covering the data, analysis, and visualization layer of our proposed Spreadsheet 2.0 architecture, and elaborates on relationships and dependencies between them. In the subsequent sections, the work at hand demonstrates the Spreadsheet 2.0 prototype based on an exemplary use-case in the domain of EAM. Furthermore, we present key findings and key issues as identified during the application of the Spreadsheet 2.0 prototype and the interviews with knowledge workers of three different companies.

The conceptual design of Spreadsheets 2.0 is based on the six defining structural capabilities of Web 2.0 technology as defined by Hertogh et al. (2011). Therefore, we addressed those capabilities as summarized

in Table 1. However, as revealed by the evaluation of the Spreadsheet 2.0 prototype and by the conducted interviews, some of the design decisions as described in the current work should be revised. In order to make a reliable assessment of the Spreadsheet 2.0 design and prototype, there is still the need for a more extensive evaluation in a business environment. For this purpose, we already established a workshop series with up to 20 industry partners from different domains which are involved in the further development of the Spreadsheet 2.0 application and its underlying Hybrid Wiki platform. In this context, we also plan to apply and evaluate the Spreadsheet 2.0 prototype.

Based on the concept and prototype as described in the paper at hand, our future research activities will focus on the design and implementation of further spreadsheet concepts (e.g., a spreadsheet-like data-grid for data editing) and Web 2.0 features (e.g., data integration) as well as their integration with the current Spreadsheet 2.0 application. Furthermore, we will examine how to further improve usability and collaborativeness of the prototype, e.g., by providing end-user-oriented features for the definition of queries or by providing real-time collaboration when designing dashboards respectively.

Web 2.0 Requirements	**Spreadsheet 2.0 Feature**
Reusability of data, functionality, and visualization	The Hybrid Wiki concepts in Figure 1 are reusable in any number of functions or visualizations, functions allow definition of reusable functionality, generic visualization types (c.f. Figure 4) are shared across workspaces and can be instantiated with domain-specific data.
Adaptability of data, functionality, and visualizations	All concepts as presented in this work (c.f. Figure 1 and Figure 4) are adaptable at runtime by end-users. Furthermore, through the analysis of dependencies between certain concepts, the system is also adoptive, i.e., automatically reacts to changes of *MxLReferables* and keeps consistency of affected *MxLDefinables* (c.f. Figure 2).
Collaborative content creation and modification	As demonstrated by the Spreadsheet 2.0 prototype, multiple and diverse knowledge workers create and edit data, its model, functions, and visualizations. An authorization model allows the specification of which users are allowed to perform which action (c.f. Figure 3).
No predefined structure of content	The emergent and end-user-driven definition of data models is the dedicated feature of Hybrid Wikis (Matthes et al. 2011).
Responsive and personalized UI	The Spreadsheet 2.0 prototype provides an interactive UI for defining functions, and visualizations. Based on the spreadsheet paradigm, it aims to provide immediate feedback to the user.
Gathering of collective intelligence	Knowledge workers with different expertises contribute to the management of the data, analysis, and visualization model within the Spreadsheet 2.0, e.g., web developers implement generic visualization types, business-users maintain domain-specific data, and programmers define complex and reusable analysis functions.

Table 1. Mapping of Web 2.0 capabilities (Hertogh et al. 2011) to Spreadsheet 2.0 features

REFERENCES

Berthold, H., Rösch, P., Zöller, S., Wortmann, F., Carenini, A., Campbell, S., Bisson, P., and Strohmaier, F. 2010. "An Architecture for Ad-hoc and Collaborative Business Intelligence," *Proceedings of the EDBT/ICDT Workshops* , pp. 13:1–13:6.

Bhat, M., Reschenhofer, T., and Matthes, F. 2015. "A Model-Based Approach for Retrospective Analysis of Enterprise Architecture Metrics," *Proceedings of the International Conference on Enterprise Information Systems* , pp. 595–611.

Bradley, L., and McDaid, K. 2009. "Using Bayesian Statistical Methods to Determine the Level of Error in Large Spreadsheets," *Proceedings of the International Conference on Software Engineering* , pp. 351–354.

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

Dayal, U., Vennelakanti, R., Sharma, R., Castellanos, M., Hao, M., and Patel, C. 2008. "Collaborative Business Intelligence: Enabling Collaborative Decision Making in Enterprises," *On the Move to Meaningful* , pp. 8–25.

Hauder, M., Matthes, F., Roth, S., and Schulz, C. 2012. "Generating Dynamic Cross-Organizational Process Visualizations through Abstract View Model Pattern Matching," *Proceedings of the International Conference on Interoperability for Enterprises Systems and Applications* , pp. 95–101.

Hertogh, S. de, Viaene, S., and Dedene, G. 2011. "Governing Web 2.0," *Communications of the ACM* (54:3), pp. 124–130.

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

Kaufmann, J., and Chamoni, P. 2014. "Structuring Collaborative Business Intelligence: A Literature Review," *Proceedings of the Hawaii International Conference on System Sciences* , pp. 3738–3747.

Lieberman, H., Paternò, F., Klann, M., and Wulf, V. 2006. "End-User Development: An Emerging Paradigm," in *End User Development*, H. Lieberman, F. Paternò, M. Klann and V. Wulf (eds.): Springer, pp. 1–8.

Matthes, F., Neubert, C., and Steinhoff, A. 2011. "Hybrid Wikis: Empowering Users to Collaboratively Structure Information," *Proceedings of the International Conference on Software and Data Technologies* , pp. 250–259.

Matthes, F., and Neubert, C. 2011. "Wiki4EAM - Using Hybrid Wikis for Enterprise Architecture Management," *Proceedings of the International Symposium on Wikis and Open Collaboration* , p. 226.

McAfee, A. 2009. *Enterprise 2.0: New Collaborative Tools for Your Organization's Toughest Challenges:* Harvard Business Press.

Mertens, M., and Krahn, T. 2012. "Knowledge Based Business Intelligence for Business User Information Self-Service," in *Collaboration and the Semantic Web*: *Social Networks, Knowledge Networks, and Knowledge Resources*, S. Bruggemann (ed.): Information Science Reference, pp. 271–296.

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

Pemberton, J. D., and Robson, A. J. 2000. "Spreadsheets in Business," *Industrial Management & Data Systems* (100:8), pp. 379–388.

Pinnington, W., Light, B., and Ferneley, E. 2007. "Too Much of a Good Thing? A Field Study of Challenges in Business Intelligence Enabled Enterprise System Environments," *Proceedings of the European Conference on Information Systems* , pp. 1941–1952.

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

Reschenhofer, T., and Matthes, F. 2015. "An Empirical Study on Spreadsheet Shortcomings from an Information Systems Perspective," *Proceedings of the International Conference on Business Information Systems* , pp. 50–61.

Reschenhofer, T., Monahov, I., and Matthes, F. 2014. "Type-Safety in EA Model Analysis," *Proceedings of the International Enterprise Distributed Object Computing Conference Workshops and Demonstrations* , pp. 87–94.

Rizzi, S. 2012. "Collaborative Business Intelligence," in *Business Intelligence*: Springer, pp. 186–205.

Schneider, A. W., Reschenhofer, T., Schütz, A., and Matthes, F. 2015. "Empirical Results for Application Landscape Complexity," *Proceedings of the Hawaii International Conference on System Sciences* , pp. 4079–4088.

Sell, D., Cabral, L., Motta, E., Domingue, J., Hakimpour, F., and Pacheco, R. 2005. "A Semantic Web based Architecture for Analytical Tools," *Proceedings of the International Conference on E-Commerce Technology* , pp. 347–354.

Spahn, M., Kleb, H., Grimm, S., and Scheidl, S. 2008. "Supporting Business Intelligence by Providing Ontology-based End-user Information Self-service," *Proceedings of the International Workshop on Ontology-Supported Business intelligence* (2008), pp. 10:1-10:12.