Private Data in Collaborative Web Mashups

Gregor Blichmann, Andreas Rümpel, Martin Schrader, Carsten Radeck and Klaus Meißner
Faculty of Computer Science, Technische Universität Dresden, Dresden, Germany

Keywords: Web Mashups, Collaboration, Private Data, Application Sharing, Impression Management, Corporate Policy Enforcement, User-driven Development.

Abstract: Situational development and utilization of long-tail Web applications often involves scenarios with multiple persons interacting. Keeping private data under control comes into focus when using applications in sensitive domains, such as financial management. Main problems comprise the lack of data restriction capabilities in an adequate granularity, missing awareness on data restricted as well as missing visual representations both on previewing such private data and replacement of non-shared data on the invitee's side. To this end, we present an innovative sharing process with fine-grained control of private data, sharing awareness and impression management. Further, policy compliance for private data enables corporate use, fostering the utilization of the collaborative Web mashup paradigm in business application scenarios. A user study shows the suitability of the corresponding interaction features for the target group of Web users with no programming skills.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

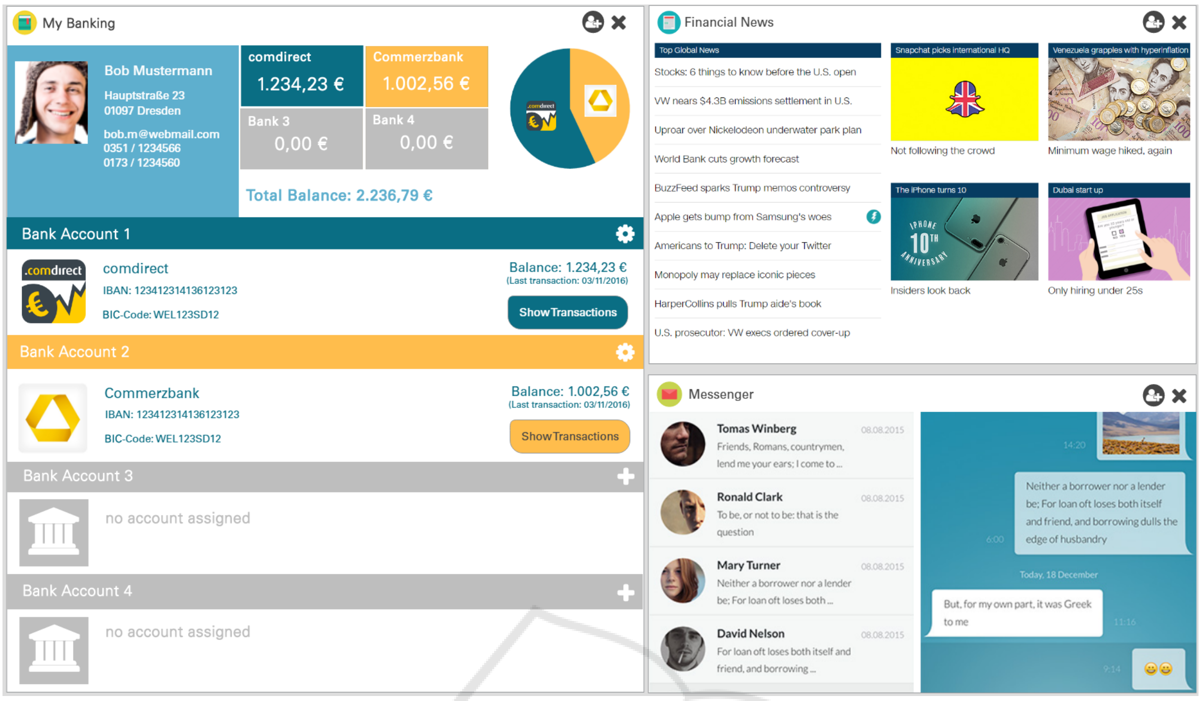
Imagine a setting with multiple Web users participating in a collaborative online session, using a platform for composite Web applications, which we call *Web mashups*. Each user may modify his individual mashup application by adding or removing application parts, i. e. *mashup components*, even at runtime. Collaboration is achieved by defining shared components, activating a partial application state synchronization. Thus, the roles *sharer* (or *inviter*) and *sharee* (or *invitee*) emerge, which both are in the target group of Web users with no programming skills. They perform situation-driven development of applications for niche purposes, also known as the "long tail".

1.1 Motivation

Private data to be protected in such collaborative Web mashups originates from several sources. On one hand, data coming into a mashup component from an external service, which e. g. contains personal account data, should be private. On the other hand, manual input data generated during application use may be confidential. It occurs in advance of the actual collaboration, when preparing the session, or during the session in non-shared application parts. The main problem is, that the sharer does not want his private data to be exposed. As a consequence, he either shares his component nevertheless and has a bad feeling or he does not share it at all, which inhibits productivity. Even worse, the sharer could not be aware of the existence and quantity of private data contained in his shared application components. Assuming a private data selection in the sharing process, follow-up problems occur on both sides: The sharer is not aware of the granularity of restricted data as well as the relationship between user interface (UI) representations and actual data. On the other side, the sharee may be confused about missing data or its visual counterparts, if no intelligent replacement, default value or data aggregation strategy is applied. Hence, on the process level, we address the challenges of private data management occurring during collaborative application use. On the UI level, we consider awareness features on both sides of the sharing invitation such as impression management. This includes control elements for revoking sharing permissions in advance and retrospectively with the objective of matching the target group's skills.

To this end, the paper's main contribution is an integrated sharing, restriction and preview process for collaboratively using composite Web mashups. It maps private data selections on a user-suitable granularity to data properties, composition fragments and capabilities in the mashup world. Additionally, cor-

Blichmann, G., Rümpel, A., Schrader, M., Radeck, C. and Meißner, K.
Private Data in Collaborative Web Mashups.
DOI: 10.5220/0006374201740183
In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017)*, pages 174-183
ISBN: 978-989-758-246-2
Copyright © 2017 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved


My Banking

Bob Mustermann
Hauptstraße 23
01097 Dresden
bob.m@webmail.com
0351 / 1234566
0173 / 1234560

comdirect 1.234,23 €
Commerzbank 1.002,56 €
Bank 3 0,00 €
Bank 4 0,00 €

Total Balance: 2.236,79 €

Bank Account 1
comdirect
IBAN: 1234123141361231123
BIC-Code: WEL123SD12
Balance: 1.234,23 €
(Last transaction: 03/11/2016)
Show Transactions

Bank Account 2
Commerzbank
IBAN: 1234123141361231123
BIC-Code: WEL123SD12
Balance: 1.002,56 €
(Last transaction: 03/11/2016)
Show Transactions

Bank Account 3
no account assigned

Bank Account 4
no account assigned

Financial News

Top Global News
Stocks: 6 things to know before the U.S. open
VW nears $4.3B emissions settlement in U.S.
Uproar over Nickelodeon underwater park plan
World Bank cuts growth forecast
BuzzFeed sparks Trump memos controversy
Apple gets bump from Samsung's woes
Americans to Trump: Delete your Twitter
Monopoly may replace iconic pieces
HarperCollins pulls Trump aide's book
U.S. prosecutor: VW execs ordered cover-up

Snapchat picks international HQ
Not following the crowd
Venezuela grapples with hyperinflation
Minimum wage hiked, again
The iPhone turns 10
Insiders look back
Dubai start up
Only hiring under 25s

Messenger

Tomas Winberg 08.08.2015
Friends, Romans, countrymen, lend me your ears; I come to ...
Ronald Clark 08.08.2015
To be, or not to be: that is the question
Mary Turner 08.08.2015
Neither a borrower nor a lender be; For loan oft loses both ...
David Nelson 08.08.2015
For loan oft loses both itself and friend, and borrowing ...

14:20
6:00 Neither a borrower nor a lender be; For loan oft loses both itself and friend, and borrowing dulls the edge of husbandry
Today, 18 December
But, for my own part, it was Greek to me 11:16
9:14 😀😀

Figure 1: Finance mashup application with the components *My Banking* (left), *Financial News* and *Messenger* (right).

porate use in enterprise scenarios is addressed by introducing mandatory and optional restriction policies on the process and UI level. Auxiliary techniques, such as view bindings, awareness features for shared and restricted application parts as well as a preview, facilitate the collaborative mashup use further, which is validated by a user study. The practical significance of the described problems and contributions is clarified in the following reference scenario.

1.2 Reference Scenario

The main research challenges of this paper are illustrated by the following financial consulting scenario. Bob manages his private financial assets using a self-developed mashup. Its start screen with three components is shown in Figure 1. *My Banking* offers four slots for adding personal bank accounts. Currently, it visualizes the balances and details of two bank accounts. The top part of the component shows Bob's contact details as well as a financial overview including the total balance of all bank accounts. In addition, a pie chart visualizes the total balance distribution on active accounts. Beyond this read-only part, each bank account is represented by a single row comprising the bank name, the bank account number and the current balance. *Show Transactions* can be clicked to list all transactions corresponding to an account. *Financial News* aggregates different news feeds to monitor current economy and finance news. *Messenger* represents an aggregation of Bob's email conversations.

Due to a friend's advice, Bob tries out a new online provider for remote financial consulting, which offers initial interviews for free. Therefore, Bob and his assigned financial advisor Harry join a video chat session. In addition, Bob starts his mashup with the intention of discussing details on his "Commerzbank" account with Harry.

At this point, the application is exclusively visible to Bob, since he did not share anything yet. After clicking at the small user icon at the top right corner of the *My Banking* component, a dialog for sharing this component which is similar to the upper part of Ⓐ in Figure 4 appears. Before Bob finishes the sharing definition, the platform detects that *My Banking* contains and visualizes Bob's personal address and bank account number. Because Bob marked both as confidential in previous sessions, it recommends to do the same now. He agrees on the recommendation and uses the sharing preview of the platform to get an impression of the data visible for the financial advisor, before finally starting the collaborative session. Thereby, he notices, that the advisor can see the balances for all of his bank accounts. He refines his sharing definition by marking one of the accounts as private and starts the sharing. Next, Harry receives an invitation to use the shared bank account component with the hint, that the component was configured to restrict confidential data in collaborative use. To avoid misunderstandings, all UI elements, which are completely or partly visualizing data model elements marked as private, are hidden and visually annotated. A small indicator is attached to the selected component, providing private data awareness for Bob. After hovering the indicator, the component UI changes and visualizes all appearances of parts of the data model which where marked as private. In addition, Bob may open the impression view again at any time, visualizing the component in the way it would be rendered for a selected collaboration partner.

After discussing Bob's situation and needs, Harry intends to offer him a financial product by sharing a product catalog with him. Due to a corporate policy of Harry's company, all information about company-specific purchasing prices is hidden from Bob. Although Bob is pleased with Harry's offers, he is pressed for time and must join another meeting. Both agree on a second consultation. Finally, Bob stores the whole session and his privacy settings as a new personal policy for reuse in the future. Simultaneously, the platform automatically creates a protocol of the consulting session, which omits all private data.

The remaining paper is structured as follows: Section 2 describes an underlying runtime architecture for using composite Web mashups collaboratively. Next, Section 3 presents the extended sharing process comprising private data restriction and policy management. In Section 4, visual interaction tools and and awareness features are discussed with regards to fit the proposed concepts to the target user group. Section 5 describes the methodology and the results of a user study. Related work is discussed briefly in Section 6. Finally, Section 7 concludes this paper by discussing the achievements of the proposed results.

2 PRIVATE DATA IN COLLABORATIVE WEB MASHUPS

To overcome the limitations of traditional monolithic groupware, such as missing interoperability and customizability, our concept adheres to the universal composition approach introduced by (Pietschmann, 2009). Thereby, arbitrary components from all application layers, including data, logic and UI, can be used to create Web mashups. Supported by recommendation techniques and visually hiding the underlying application complexity, users with no programming skills are empowered to iteratively build and customize desired app functionalities while using those mashups. Thereby, components are black boxes and encapsulate arbitrary Web resources like data feeds, Web services or UI widgets. To establish an orchestrated data flow between components from different third-party providers, all components implement a public interface comprising operations, events and properties. Properties represent the component's state utilizing semantically typed key-value pairs. Optionally, component-specific default values, which can be overridden by the application's configuration, support initialization routines. State changes, indicated by events, can be used to invoke operations by parameterized messages. Such communication interfaces as well as the component's functionality may be annotated with semantic *capabilities*, which facilitate user-driven composition and data type mediation. Therefore, concepts of third-party ontologies are used to increase interoperability of different components. As a basis functionality for awareness features, interface elements, such as properties, may be assigned to a set of UI representations by an additional annotation. These *view bindings* make use of Cascading Style Sheets (CSS) selectors to address corresponding visual component elements, such as a label, a diagram or an input field, in a flexible way.

On top of that paradigm, (Blichmann et al., 2016) introduced an approach, which enables users of the target group to define and evaluate individual sharing definitions by simply managing visual triples of *who*, *what* and *how*. These triples can be used to share arbitrary applications, single components or even parts of them with different collaboration partners for synchronous use during runtime. Within such a dynamic scenario, empowering the user to manage his confidential application data is a challenging task. To address the target group of non-programmers, configuration and technical details have to be hidden and automated preferably. While several standard technologies exist to, e. g., ensure transport encryption (TLS) or protecting user accounts (WebID), protecting confidential application data is highly specific to a certain domain context. This can only be achieved by involving the user. Hence, this paper focuses on protecting confidential application data, which occurs situationally or is determined due to corporate policies.

3 DEFINING RESTRICTIONS ON PRIVATE DATA

For sharing mashup components while preserving the control over private data, we define restrictions on private data as an extension to the process of sharing application components. Further, we describe, how policies are enforced as mandatory restrictions and how they influence user perception and functionality.

3.1 Sharing with Restrictions

The first requirement on restricting private data is user-driven, cf. scenario in Subsection 1.2. Bob wants to share his “Commerzbank” account only, but the default share action is based on the complete banking component and synchronizes its entire state. Each different bank account is represented as a single component interface property which in total represent the current inner state of the component. However, bank accounts are represented as complex data type, which is described via an ontology and includes for example the name of the bank, the bank account number and the account balance.

Initially, Bob defines a sharing triple, cf. Section 2, with the bank account component as the sharing object, Harry as a single user subject and the *how* part assigned to *view*. The original process would now apply the application sharing by inviting Harry to view the *My Banking* component. Our proposed sharing process provides extensions during the definition of the sharing triple as well as a preview mode implementing *impression management*, which allows to view the application through the eyes of another user. The extensions are highlighted in Figure 2, which illustrates the advanced sharing process.

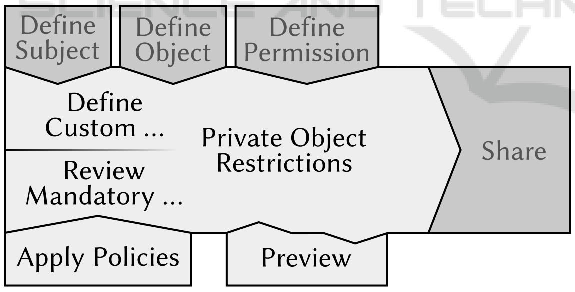


Figure 2: Process of defining shared objects with restrictions, policies and a preview option.

Overall, the platform allows to define three different types of private data objects depending on the previously selected sharing object. If a user selects the whole application to be shared, entire components, single application features or application data can be restricted. Application features are represented by the component’s capabilities which are linked to the properties of the component, cf. Section 2. Thereby, the user may select either application features like *Display bank account details* or application data like the aforementioned bank accounts. Such data is generated from a list of all component properties and there types. If only a single component is intended to be shared, cf. Subsection 1.2, users can only specify application features or data of that component to be private. Similarly, if the sharing definition specifies a set of application features, only the including application data can be restricted.

To restrict only bank account numbers as parts of shared component properties, property types have to be decomposed to be provided for UI selection. Regarding the reference scenario, Bob therefore is now able to either select all data elements of bank account “comdirect” or to specify only the corresponding bank account number to be private. For type decomposition, the semantic annotation of each property is used. Therefore, provided sub types are extracted from a domain ontology. This implies a more flexible selection of restricted objects and thus creates independence from the granularity of properties defined by the component author. To restrict the visualization of a certain bank account at all, Bob selects the corresponding property to mark it as restricted. To do this, he first has to select subject, object and access permissions. A user may not decide freely about which properties to restrict or leave unrestricted, since a *policy* is able to narrow down the space of options. Applying such policies is discussed in Subsection 3.2. Assuming a rather complex sharing restrictions setup, the user is offered a preview of settings applied. The impression management provided by this preview mode is described in Subsection 4.3. After leaving or turning down to use the preview mode, the actual sharing process is initiated.

3.2 Policies

Since favorites and templates on private data restrictions are required by the scenario, we introduce configurable policies, which define sets of such restrictions for certain purposes. The structure of policies is defined in a metamodel in Figure 3.

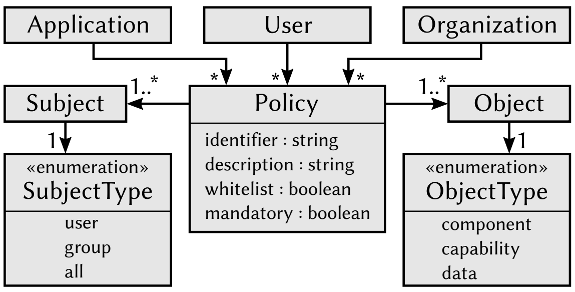


Figure 3: Policy metamodel.

A *mandatory* policy occurs when Harry has to respect corporate rules, cf. scenario. Such policies cannot be overridden. Thus, only awareness information on applying the policy is required. Mandatory policies are also useful for enforcing legal requirements. Policies can also have a *recommendation* function, e. g. when Bob is applying favorite settings from previous consulting sessions. In that case, an additional selection or confirmation step is needed.

Policies appear in different scopes. First, a *corporate policy* repository serves mandatory and optional policies within the scope of an organization or organizational unit. Thus, corporate rules may be enforced, but it is also a platform for recommending best practice policies to other colleagues in the company. In a corporate repository, each user is allowed to create policies, also for other single users or groups. These policies have a recommendation function. Corporate policies with the mandatory setting activated require a special administrative role to be created and assigned. On an *individual level*, each user should be able to save his favorite policies with private visibility. In this case, the policy is attached to the user's profile, which is can be used in arbitrary application contexts. When defining domain-specific policies, they are best stored in *application scope*. This way, all users of a certain application may profit from the application specific policy, since it is attached to the application model.

While common data restrictions on shared components are described in a *blacklist* fashion, i. e., the specified objects are blocked, also *whitelist* policies can be specified. They develop their full potential in application-agnostic cases, e. g., when you want to define, that nothing but a specific set of properties is allowed to be shared.

Each policy is assigned to at least one subject, which can be specific users, a group of users, which facilitates assigning permissions to certain roles, or all users in general. In the example scenario, Harry is member of the group *bank employees*. Finally, the objects to be restricted are defined by listing the data properties, cf. Section 2, application composition fragments of a super fragment shared or capabilities of such composition fragments.

4 USER INTERFACE AND AWARENESS

The extended sharing and restriction process comes with integrated visual tooling support, which is based on the sharing UI presented in (Blichmann et al., 2016). This section describes the ad-hoc specification of private data, the awareness features in live view and the impression management for reviewing applied settings.

4.1 Ad-hoc Specification of Private Data

The ad-hoc specification of restrictions on private data is an extension of the triple-based sharing dialog, cf. Section 2. As indicated in the top left of Ⓐ in Figure 4, users visually compose a triple of *who*, *what* and *how*. Since users may specify triples in an arbitrary order, the dialog for selecting private data appears as soon as each of the three parts is assigned to at least one user, composition fragment or permission respectively. In the lower part of Ⓐ, the blue privacy selection panel visualizes all component parts, i. e. component data and features, in accordance with the concept for selectable private objects presented in Subsection 3.1. They are grouped by the components selected at *what*. Per default, all data elements of a component are displayed as visible or editable, depending on the overall permission selected at *how*. Thus, the permission icons from the *how* selection are reused. Each data element, i. e. property tag, can be toggled private by clicking on it. This changes the tag's appearance by replacing the icon with a *slashed eye*, which indicates, that it cannot be seen by the invitee. This visualization, e. g. shown for *bank account 1* in Figure 4, was introduced as a result of the user study. The previously used lock icon caused some confusion, see Subsection 5.3. To create more fine-grained restrictions, like the bank account number of *bank account 1*, users can open a drop-down view by clicking the arrow icon on the right side of each data element. In the presented case, a user can switch between the selection of data or features by using the corresponding tab panel.

Another requirement identified by the user study, is a clear visual separation between user-selected data and mandatorily locked data defined by a system policy. As an example in Ⓐ, *personal address* and *account ID* are marked as private by a mandatory policy. Therefore, they are visualized with a striped red background. In addition, a lock icon now is used to clarify, that the user is not allowed to toggle the element. Further assistance is provided by displaying context-sensitive advices and explanations, depending on the currently activated step, see right column with light background in Ⓐ. In the depicted step, the figure shows a motivation message on private data specification, its intended interaction behavior and an explanation on policies and their tag appearance.

Assuming the correct interpretation of the presented data element labels is difficult for some users, they can use the question mark icons attached directly to each data element. After clicking, the user is forwarded to a customized live preview. It highlights the corresponding UI fragments which would be affected

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

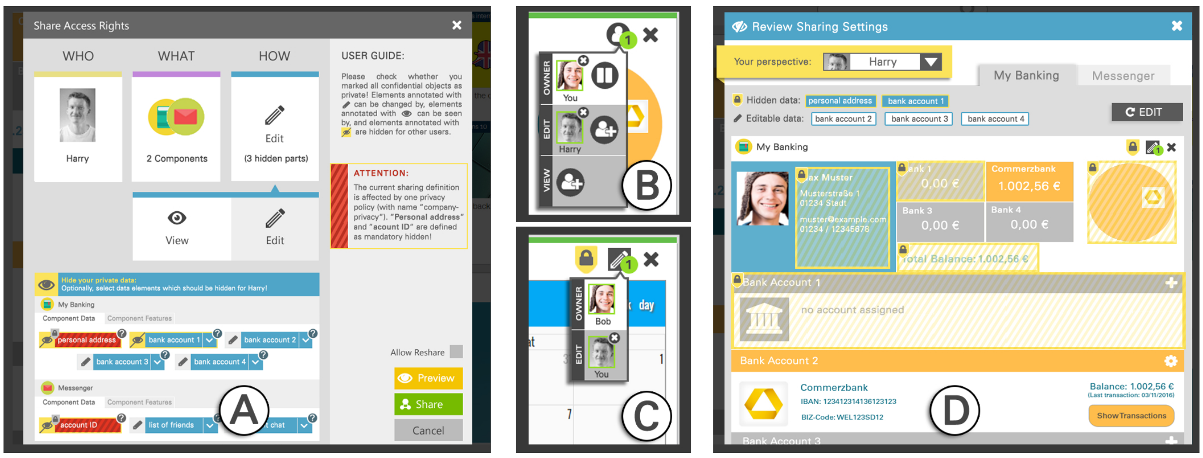


Figure 4: Ⓐ private data restriction and policy visualization, Ⓑ + Ⓒ live view awareness and Ⓓ impression view.

by toggling the corresponding data element. To ensure consistency, the preview panel is realized by the impression view, which will be described in detail in Subsection 4.3. The green share button, see lower right corner of Ⓐ, accepts the sharing definition by sending invitations to the selected users.

4.2 Live View Restriction and Awareness

Depending on the user's role as either inviter or invitee, the application's live view changes as indicated at Ⓑ and Ⓒ in Figure 4. When Bob applies sharing configuration by clicking the *share* button at Ⓐ, all shared components are augmented with a green line at the top of the tool bar, see Ⓑ. Next, the sharing icon changes and now indicates the number of invited users. By clicking on this icon, a compact sharing overview is activated, which lists all members of the collaborative group with access to the selected component. Each user is represented by a profile image, whereas users are visually clustered according to their permission to edit or view. A green border around a profile image indicates, that this user is currently online and active. This behavior was also introduced as a result of the user study, cf. Subsection 5.3. According to that, users prefer to review detailed access permissions by clicking on the profile image instead of a dedicated restriction symbol, such as a lock icon. To address this, the impression view Ⓒ, which is described in detail in Subsection 4.3, opens as modal window. Next to the small profile icons in each swim lane, the sharing icon known from the component's title bar, cf. Figure 1 provides a shortcut to the sharing definition dialog shown in Ⓐ. It opens with preselected values for *what* and *how*. To exclude single users or to stop the synchronization of this component at all, the inviter can use the small close icons attached each user image. The pause icon visualized by two vertical bars, at the top right can be used to temporarily pause the propagation of state changes to other users of the collaborative session.

If an invitee accepts an invitation, the appearance of the shared components is similar. Additionally, a protection shield icon at the top bar of a component helps to indicate that the component's UI is partly blocked due to privacy settings. When data is marked as private by the component's owner (the inviter), it is prevented from being synchronized and thus, it is replaced with default values, which cannot be changed by the invitee. This default data is extracted from the component interface description, which is provided by the corresponding component developer. Having this replacement strategy with default values offers the advantage of preserving the layout instead of deleting DOM nodes with confidential data. As an awareness feature, all DOM nodes representing private data are visualized by a striped overlay. Additionally, they are surrounded by a yellow border and contain a yellow and black lock shield icon. To identify the corresponding UI representations appearing as DOM nodes, the component's view bindings and capability annotations are used to select these elements with the help of the annotated CSS selectors. When a user toggles a certain data element of a component, such as the bank account 1 of *My Banking*, to private, the platform has to consider two cases. First, the view bindings, which are directly attached to the capabilities of the property representing the data element, are used to identify and format the referenced DOM nodes as discussed above. Second, all capabilities, which are modeled interdependently in the mashup component model using *caused by* or *causes* relations, are blocked as well. The latter is required to ensure, that data marked as private is not visualized within another context or as part of combined data set, like for example the pie chart of the *My Banking* component. Even if the invited user has the permission to interact with the component, parts marked as private cannot be changed, even if they are visualizing default data values. Invitees are notified about restricted data through visually annotating corresponding UI elements to ensure that both sides has the same context for their collaborative task. Hiding the general existence of data elements and treating associated problems is out of scope of this paper.

An invited user can also open the comprehensive permission overview, see Ⓒ, to see all users, which have access to this component. However, the invitee is not able to open the impression view by e. g. clicking at the user image of the owner or any other user to see some details within his view. The invitee is only able to terminate his own participation in the collaborative session, not the sessions of other users.

4.3 Impression View

To ensure that users interact with high confidence concerning the security of their data, the system enables to simulate the live view perspective of other users. The impression view can be seen in action in Ⓓ, presenting the perspective of Harry, which was invoked by Bob to ensure, that the desired bank account is blocked. Thus, it provides a preview for Bob showing the component's UI as seen through the eyes of Harry. We introduced this technique due to positive experiences provided from different literature like (Leary and Kowalski, 1990) and due to multiple user requests, which came up in the study.

The impression view fulfills two tasks. Inviters may check the settings of a new sharing definition before sending the invitation to other users. Having already shared components, it serves as a review of active sharings by clicking at the user icon in the small sharing overview panel, cf. Ⓑ.

Within the view, users first have to select the desired user perspective by a drop down selection box placed within the yellow area at the top of Ⓓ. The selection list includes all users which received permissions for this component from the current user. To keep the UI clear and simple to understand, only one component is visualized at the same time. If the selected user has access to more than one component, he can switch between them by using the tabs at the top. Within a tab, all data elements of a component are grouped according the possible permissions to be hidden (private), editable or only visible. To be consistent, the icon set of Ⓐ is reused here. In the example of Figure 4, personal address and bank account 1 are visually blocked. The other data elements can be visually manipulated by Harry. However, there is no differentiation for data blocked due to policy rules or individual privacy definitions. The impression view's body visualizes the component in the way, the corresponding user perceives it. Therefore, it employs the visualization techniques described in Subsection 4.2. When users detect unintended configurations or changed their requirements, they can use the edit button to switch to the panel presented in Ⓐ. Thereby, the panel gets initialized with the current sharing configuration and the label of the share button switches to "update". In addition, a button is attached to completely revoke the sharing.

5 EVALUATION

The mashup-based concept of managing private data with restrictions and policies is evaluated with regard to its practicability by a reference implementation within the CRUISE runtime environment (Subsection 5.1). To test the user acceptance, a user study was conducted (Subsection 5.2 and Subsection 5.3).

5.1 Implementation

The CRUISE platform hosts mashup applications to manage private data on. Therefore, a server-side coordination layer is implemented as a singleton Enterprise Java Bean (EJB) for all clients. The client side is realized by standard Web technologies (HTML5, CSS and JavaScript). Client-server communication is done by Web sockets using Apache Apollo[1]. The representation of private data is realized as an extension of the access control list (ACL) implementation presented in (Blichmann et al., 2016). Thereby, the ACL on the client side is represented via JSON and on the server side via Java. To store new private data settings, the system creates a new sharing triple within the ACL. All targeted user ids are marked as subject. The set of intended private properties is stored as object. The permission part is extended by a privacy option, which is activated in that case. Similarly, a set of triples is created to implement a policy definition. Updates are exchanged between client and server by dedicated commands to keep the ACL consistent for all clients.

[1]https://activemq.apache.org/apollo/

5.2 Methodology

To evaluate the user acceptance of the proposed process of defining private data, usability testing with think-aloud protocols was performed. Test users had to pass a tripartite setup. First, an introduction on how to use and modify composite Web mashups as well as on the reference scenario presented in Subsection 1.2 was given. Second, participants were asked to complete four tasks with increasing complexity by using interactive paper mockups, which were created with the help of the click prototype tool *invision*[2]:

- Share the banking component with the permission to interact, restrict the “comdirect” account and ensure that it is restricted properly.
- Share the banking and messenger components with an activated restriction policy, identify and explain the influence of the mandatory restriction. Discover the preview here or in the first task.
- Review the shared data from the invitee’s point of view, but still logged in as the sharer.
- Explain sharing awareness features logged in as the invitee and exemplify permitted and restricted UI actions.

A moderator recorded all comments. After each individual task, he updated the prototype setup. To get a comparable and standardized result, in the final step, the participants were asked to fill in the System Usability Scale (SUS) questionnaire.

The user study was conducted with the help of five male and seven female participants including ages from 24 to 33 years with an average of 29 years. Ten users had no programming skills but frequently use the Web. Thus, they fit the intended target group best. Professions were widely spread, including engineering, chemistry, linguistics, fine arts and architecture.

5.3 Results

The overall feedback of all participants was very positive. Starting the creation of a new sharing definition by selecting subject (*who*) and object (*what*) caused no problems, but many participants asked for drag and drop support. Since the selection of the overall permission (*how*) was shown in parallel to the privacy data selection, some users skipped the step of actually selecting a permission in the first place. Inspired from that, the current process requires all three basic sharing triple elements to be assigned, before the restriction UI area appears. Fortunately, the general idea of the private data selection approach was clear from a semantic point of view.

Within the restriction of private data elements, some of the participants disliked the initial visualization of not selected data elements. Some participants interpreted them as not be shared at all, since they appeared as gray boxes. To solve this, the current concept provides the uniform shape and color for all data elements and just changes the attached permission icon after clicking, see Ⓐ in Figure 4. Other participants expected to configure the shared visibility of a bank account directly in the live view instead of first defining a component to be shared. In most cases, both private data selected by users and those originating from a mandatory policy were interpreted correctly. However, users expected the visual differences of those two types to be more notable. Only in a few cases, the participants expected the mandatory policy selection to be modifiable, which might be caused by the corresponding hint being not obvious enough. Thus, we exchanged the mandatory policies with a striped background shape in red color and emphasized the visual link to the corresponding hint text. Help icons were accepted without problems and considered helpful indeed. The impression view was considered very helpful. Some users demanded such a feature even without recognizing the preview option in Ⓐ of Figure 4. Overall, most of the users were very confident of having chosen their private data correctly. Within the impression view, awareness functions were interpreted correctly, even the locks on visual elements corresponding to aggregated data, being under the influence of restricted data.

Within the initial concept, inviters and invitees were allowed to activate a dedicated review mode by clicking on a lock icon, which was shown at the top bar of the component. This was rated as not very intuitive for many participants, since they did not expect anything to happen, when clicking on a lock. We incorporated this functionality into the impression view. It is activated by clicking the image of a collaborating user, as demanded by the study participants.

In general, we observed, that self-descriptiveness is a very important quality of Web systems in modern society. Almost all participants ignored headlines, hints, and long text descriptions. They prefer the paradigm *exploration before reading*. All participants filled in the SUS questionnaire and created an average score of 75.4, which indicates a good value for the general usability of the evaluated platform interaction. Individual results are listed in Table 1.

[2]https://www.invisionapp.com/

Table 1: Characteristics and SUS ratings of study participants.

Age	Sex	Profession	Programming Skills	Web Usage	SUS Rating
31	♂	engineering	none	daily	67.5
31	♂	construction engineering	none	daily	70.0
33	♂	computer science	professional	daily	80.0
25	♀	fine arts student	almost none	daily	75.0
24	♀	chemistry student	none	daily	92.5
28	♀	management assistant	none	frequent	77.5
27	♀	economy student	none	daily	72.5
29	♀	linguistics student	none	daily	67.5
30	♂	philosophy	none	daily	70.0
27	♀	linguistics student	none	daily	80.0
29	♀	architecture	none	daily	70.0
32	♂	software engineering	professional	frequent	82.5

6 RELATED WORK

This paper's research challenges align themselves into the areas of collaborative, multi-device mashup environments, personal learning environments (PLEs) and co-browsing, which provide alternative solutions. *PEUDOM* (Picozzi, 2014), enables non-programmers to share and synchronize mashup applications during runtime with the help of a client-server platform. Its permission management neither covers any UI support, nor considers private data or application parts during a collaborative session. In contrast to PEUDOM, the approach of Tschudnowsky et al. (Tschudnowsky et al., 2014) uses choreographed mashups. Utilizing operational transformation (OT), they focus on concurrency control mechanisms, but again offer no support for privacy or private data.

MultiMasher (Husmann et al., 2013), a multi-device mashup approach, supports collocated scenarios where collaborators can synchronize parts of their personal mobile devices with one publicly shared screen. They are able to select parts of an arbitrary DOM-based Web site, isolate each of them as a component and migrate it during runtime between their personal, hidden device and the public screen. However, it is only possible to hide components entirely. Hiding single parts of its data model and using policies to reuse privacy settings is not possible.

Sqwelch (Fox et al., 2011) allows composing mashups in the domain of digital health records based on a trust level, which is specified by users. A social network is used to manage the relationship of trust between medical professionals, care givers and patients as a foundation to ensure data privacy. However, the presented features address only coordination and not real-time collaboration scenarios. In addition, trust can only be given to or revoked from users or complete widgets. Defining widget-independent data elements or parts of the widget is not possible.

The Privacy Specification and Enforcement Model for Mashups (Chun et al., 2013) aims to protect private data, while being used by mashups. The concept includes three types of privacy policies, which are interpreted by a privacy protection engine to ensure that the privacy goals of the data provider, the mashup provider as well as the mashup user. While this delivers basic concepts for application-independent privacy specification, e. g. by using generic data types and behavior descriptors, it misses situational support for non-programmers, who need to configure and review their privacy settings at runtime.

Graasp (Bogdanov, 2013) enables to create and share "spaces" of resources and widgets. It offers no fine-grained specification of private data, but only enables application parts to be private at space level. *CURE*, a similar PLE, facilitates the key metaphor to grant access permissions to spaces, called "rooms" (Schümmer et al., 2005). CURE allows creating and distributing virtual keys during runtime and enables the users to mark single resources like documents as shared or private. Instead of interactive content, the concept only allows sharing simple assets, such as documents and videos. The approach lacks concepts for keeping parts of shared documents private.

CoCAB (Niederhausen et al., 2010) empowers remote consulting sessions within a co-browsing environment. Assuming a Web page contains form-based UI elements, users can specify single form fields, such as passwords, to be private. The "form protector" can differentiate such fields as publicly editable, only viewable or completely visually blocked. However, the concept is restricted to blocking single form fields. It does not consider advanced interaction forms as well as application-independent policies.

Overall, none of the presented approaches enables non-programmers to define arbitrary parts of their application as private and re-use theses settings in application-independent policies.

7 CONCLUSION

Using situational Web applications collaboratively is intuitive and easy, when facilitating the mashup paradigm with appropriate components. Platforms for user-driven development provide individual application tailoring, even for users with no programming skills. However, the analysis of related work showed significant deficiencies in the restriction of private data. This includes missing concepts for data processing as well as UI and awareness support. Thus, users, who want to collaborate, are left in incertitude about their private data or refuse collaboration at all.

This paper introduces a concept of restricting private data on shared applications parts in a Web mashup infrastructure. To this end, any user is enabled to define restrictions on private data based on an integrated tooling, which provides advanced preview features, such as impression management in advance of the actual sharing. On top of that, flexible policy constructs facilitate sophisticated scenarios, such as corporate rules. The suitability of this approach for users with no programming skills is validated by a user study including twelve participants from different professions. The major part was based on a think-aloud test. Its results yielded very positive feedback and an average SUS score of 75.4, which we found to be a very promising outcome. The feedback of the user study is already considered within the improved concept discussed in this paper.

Future work includes the projection of the concepts to other application domains and scenarios. This could validate the practicability within an even more general audience. Currently, private data can only be specified by its type. It is not possible to select specific data instances within a list of equally-typed data, like one single message within a conversation of the *Messenger*. To overcome this limitation, a concept for presenting and marking arbitrary parts of a generic data set instance is required which is appropriate for the target group of our platform. Nevertheless, we believe, that the proposed concept is a key factor for the acceptance of Web-based collaborative platforms in real life scenarios. Protecting private data and providing suitable visual confidence to all participants of a collaborative session is a crucial requirement which suchlike platforms have to cope with.

ACKNOWLEDGEMENTS

Gregor Blichmann, Andreas Rümpel, and Carsten Radeck are funded by the European Regional Development Fund and the Free State of Saxony.

REFERENCES

Blichmann, G., Radeck, C., Starke, R., and Meißner, K. (2016). Triple-based sharing of context-aware composite web applications for non-programmers. In *Proceedings of the 12th International Conference on Web Information Systems and Technologies, WEBIST 2016, Volume 2, Rome, Italy, April 23-25, 2016*, pages 17–26. SciTePress.

Bogdanov, E. (2013). *Widgets and Spaces: Personal & Contextual Portability and Plasticity with OpenSocial*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne (EPFL).

Chun, S. A., Warner, J., and Keromytis, A. D. (2013). Privacy policy-driven mashups. *International Journal of Business Continuity and Risk Management*, 4(4):344–370.

Fox, R., Cooley, J., and Hauswirth, M. (2011). Collaborative development of trusted mashups. *International Journal of Pervasive Computing and Communications*, 7(3):264–288.

Husmann, M., Nebeling, M., and Norrie, M. C. (2013). Multimasher: A visual tool for multi-device mashups. In *Current Trends in Web Engineering - ICWE 2013 International Workshops ComposableWeb, QWE, MDWE, DMSSW, EMotions, CSE, SSN, and PhD Symposium, Aalborg, Denmark, July 8-12, 2013. Revised Selected Papers*, volume 8295 of *Lecture Notes in Computer Science*, pages 27–38. Springer.

Leary, M. R. and Kowalski, R. M. (1990). Impression management: A literature review and two-component model. *Psychological Bulletin*, 107(1):34–47.

Niederhausen, M., Pietschmann, S., Ruch, T., and Meißner, K. (2010). Web-based support by thin-client co-browsing. In *Emergent Web Intelligence: Advanced Semantic Technologies*, Advanced Information and Knowledge Processing, pages 395–428. Springer London.

Picozzi, M. (2014). *End-user Development of Mashups: Models, Composition Paradigms and Tools*. PhD thesis, Italy.

Pietschmann, S. (2009). A Model-Driven Development Process and Runtime Platform for Adaptive Composite Web Applications. *Technology*, 2(4):277–288.

Schümmer, T., Haake, J. M., and Haake, A. (2005). A metaphor and user interface for managing access permissions in shared workspace systems. In *From Integrated Publication and Information Systems to Virtual Information and Knowledge Environments, Essays Dedicated to Erich J. Neuhold on the Occasion of His 65th Birthday*, volume 3379 of *Lecture Notes in Computer Science*, pages 251–260. Springer.

Tschudnowsky, A., Hertel, M., Wiedemann, F., and Gaedke, M. (2014). Towards real-time collaboration in user interface mashups. In *ICE-B 2014 - Proceedings of the 11th International Conference on e-Business, Vienna, Austria, 28-30 August, 2014*, pages 193–200. SciTePress.