Sharing Composite Web Applications with Multiple Coupling Levels

Gregor Blichmann(✉), Christopher Lienemann, and Klaus Meissner

Technische Universität Dresden, Dresden, Germany
{gregor.blichmann,christopher.lienemann,klaus.meissner}@tu-dresden.de

Abstract. Developing and utilizing situational long-tail Web applications for scenarios with multiple persons interacting gains increasing importance. Thereby, adjusting the style of coupling during runtime is necessary to support changing situational requirements and long-tail collaborative situations. Main problems comprise the lack of universal coupling levels which are characterized in a sufficient and understandable granularity, missing analyses for potential synchronization points in the context of black-box-based Web applications, and an adequate user interface (UI) support for managing and reviewing individual coupling levels for multiple users simultaneously. To this end, we present a novel concept for managing different levels of coupling within composite Web applications. The seven proposed sharing modes support Web users with no programming skills by a default configuration of couplings, permissions and sharing objects. A wizard-like interaction concept enables these users to create and manage personal sharing definitions during runtime while considering various levels of coupling for each collaborative partner. A first user study provides preliminary evidence about the acceptance of the sharing modes and the interaction features for the target group.

Keywords: Mashup · End-user development · Collaborative work · Permission management · Coupling levels

1 Introduction

Approaches for universal composition like CRUISe [1] or mashart [2] allow for a platform-independent modeling of composite Web applications (CWAs) and a uniform description of components spanning all application layers. Imagine a setting with multiple users collaborating in an online session by utilizing a platform for CWAs. Each user has an individual set of components as part of a shared application instance and can modify this by adding or removing, e.g., *mashup components*, even at runtime. Collaboration is achieved by defining shared components, which activates a partial application state synchronization. Thus, the roles *inviter* and *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".

C. Gutwin et al. (Eds.): CRIWG 2017, LNCS 10391, pp. 15–31, 2017.
DOI: 10.1007/978-3-319-63874-4_2

1.1 Motivation

Using groupware synchronously in a distributed group implies a certain coupling level. Several works (e.g. [3,4]) rated a flexible coupling support as essential for supporting multiple application purposes and changing tasks during runtime. Coupling levels can vary significantly. Tight coupling, as used in screen and desktop sharing applications like TeamViewer[1], proposes identical screens to all participants and synchronizes changes in a strict What You See Is What I See (WYSIWIS) manner. Loose coupling, for example used in Google Docs[2] or Zoho Docs[3], enables users to view different parts of a shared text and individually rework different paragraphs. Multiple coupling levels in parallel can be desired in various collaborative use cases and therefore require sophisticated support for awareness as well as coordination. Traditional, collaborative Web applications are monolithic and suffer from a predefined, restricted functional scope as well as a mostly fixed level of coupling. In this paper, we present a platform for the situational development of collaborative, composite applications for arbitrary collaborative use cases. Users with no programming skills can search, integrate and share any application part in the form of components according to their current context during runtime. The platform supports distributed mixed-focus collaboration [5], in which distributed users permanently switch between working individually and together.

This causes a major research challenge for this work. Since participants' requirements and tasks are strongly varying during runtime, suitable couplings can not be predicted or hard-coded. Users have to understand effects of the current coupling level for all parts of their application and be able to change the level depending on the current work context on their own [3]. Although approaches like [3,4] are present, no generic classification of possible coupling levels for arbitrary application and collaboration purposes exists. In the context of CWA, which synchronizes black-box components by their public interface, analyzing coupling possibilities is another challenge of this work. Suitable awareness in the context of mixed-focus collaboration is already well studied for professional users (e.g. [5]). However, our target group includes daily Web users without programming skills. For them, necessary configurations have to be automated and technical details to be hidden. Managing multiple coupling levels for different group members by the runtime environment is also a challenge tackled by this paper.

Since the presented challenges for introducing multiple levels of coupling for collaborative CWAs are not covered by any work yet, this paper presents three contributions: First, a deep analysis of different possible levels is introduced. This mainly considers the synchronization of situational, collaborative CWAs assembled by black-box components of various vendors. On top, non-programmers, as our primary target group, will be empowered to change coupling levels during runtime supported by a set of preconfigured sharing modes. An adequate UI support simplifies selection and understanding of different levels for them. Finally,

[1] https://www.teamviewer.com.
[2] https://docs.google.com.
[3] https://www.zoho.eu/docs/.

a prototype and a user study prove the user acceptance and sustainability of the concepts. The practical significance of the described problems and contributions is clarified in the following reference scenario.

1.2 Reference Scenario

The following travel planning scenario illustrates the main research challenges of this paper. Bob wants to collaboratively organize his next round trip with his friends Charlie and Alice, both currently on a business trip. Since they did not agree on a destination, Bob initiates a new mashup including three map components. Each map visualizes one of his favorite places in Spain, Italy, and Canada. To discuss his proposals, Bob shares the application by using the tightly coupled presentation mode with Charlie and Alice. Thereby, all collaborative partners will see the viewport of Bob. This includes the synchronization of his basic application layout and its changes as well as the synchronization of all his scroll and zoom activities. Charlie and Alice can only consume Bob's changes and use the integrated video chat to bring in their ideas.

After they selected Canada, Bob removes the other maps and adjusts the sharing configuration for Charlie and Alice to a tightly coupled collaboration mode. Thereby, all users can change the components' content, for example by adding markers or change the visual appearance, for example by changing the map type. Due to this coupling level, both kinds of changes of all users are synchronized with all members of the group.

Some discussions later, the map contains a set of markers representing the desired places of the planned round trip. To agree on the date of the trip, Bob adds a calendar component to the application and shares it with the other both. He selects a coupling level, which synchronizes only the calendar's data elements, like created appointments. All UI specific configurations, like the currently visible week, can be adjusted individually by each group member. Using this level of coupling, each of the three friends can scan for available dates separately and as soon as one of them creates or adjusts the appointment representing their trip, it is visible for everybody.

Finally, Alice integrates a notepad component. She adds all necessary tasks to be achieved before starting the tour, like booking flights or reserving hostels. Afterward, she shares the component with the other both and uses a coupling level, which enables them to choose from a set of alternative notepad components. After accepting the invitation, Bob and Charlie can work with an individual notepad while the platform synchronizes the underlying list of tasks by facilitating the semantic annotations.

The remainder of this paper is structured as follows: Sect. 2 briefly introduces a runtime environment for collaborative, composite Web mashups. Next, Sect. 3 discusses possible coupling levels and sharing modes and presents their integration into the process for sharing composition fragments during runtime. In Sect. 4, visual interaction tools and awareness features are discussed with regards to fit the proposed concepts to the target user group. Section 5 describes the underlying prototype, the methodology and the results of a user study. Related

work is discussed in Sect. 6. Finally, Sect. 7 concludes this paper by discussing the achievements of the proposed results.

2 Collaborative Usage of Composite Web Applications

Traditional monolithic groupware systems lack interoperability and customizability. To overcome these limitations, our concept adheres to the universal composition approach introduced by [1]. So-called compositeWeb application (CWA) combine arbitrary components from all application layers, including data, logic, and UI. By utilizing recommendation techniques and visually hiding the underlying application complexity, users with no programming skills are empowered to build and customize desired app functionalities during runtime iteratively. Different third-party vendors develop the application's components, which encapsulate arbitrary Web resources like data feeds, Web services or UI widgets as black-boxes. To establish an orchestrated data flow between components, all of them implement a public interface comprising operations, events, and properties. Properties represent a part of the component's inner state utilizing semantically typed key-value pairs. State changes, indicated by events, can be used to invoke operations by parameterized messages. Such interfaces as well as the component's overall functionality may be annotated with semantic *capabilities* to facilitate user-driven composition and data type mediation. Therefore, concepts of third-party ontologies are used to increase interoperability of different components. As a basic 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, flexibly. A composition model represents all application-specific elements, like the included components, their communication channels, layout, and screen flow.

On top of that, the work of [6] enables users of the target group to define and evaluate custom sharing definitions by just 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 simultaneous use during runtime.

3 Levels of Coupling for Composite Web Applications

To present coupling levels for CWAs, the terms shared session, sharing definition, coupling, and synchronization have to be clarified within the scope of this work. A shared session defines a set of sharing definitions. A sharing definition potentially includes arbitrary triples with not less than one user, one composition fragment, and one permission each. A user can define multiple sharing definitions during runtime by using the sharing view of the platform. As soon as an invitee accepts a proposed sharing, its components are coupled and will be

synchronized each time a state change appears according to the corresponding level of coupling and permission.

In the following, Subsect. 3.1 discusses potential coupling levels in the context of composite mashup applications. Furthermore, Subsect. 3.2 introduces seven sharing modes and Subsect. 3.3 aligns them with the existing sharing process.

3.1 Access Options and Coupling Levels of Composite Web Applications

In contrast to traditional, monolithic groupware, CWAs contain black-box components which exchange data using semantically annotated interfaces. Therefore, it is necessary to analyze potentially suitable coupling levels and the possible access options within suchlike applications. As indicated in Fig. 1, access options for defining different levels of coupling originate from three abstraction layers. The *platform* layer comprises all runtime environment and device specific features, which are independent of an application instance. This includes, e.g., the current *viewport* present to the user, the set of received *recommendations* for additional components or application reconfigurations, or the *list of users* currently online. The *composition* represents all application-specific characteristics specified by the composition model. This includes *screen flow*, *layout*, the *set of components*, their data flow (*communication*), and its *adaptivity* behavior. The *component* layer, due to the black-box paradigm, contains all component-specific elements in form of component interface parts like properties, operations, and events. Since properties can represent arbitrary parts of a component's inner state, it is necessary to analyze their semantics in more detail. Practically, properties can represent basic *configurations* necessary to instantiate the component, like name, size, the URL of the backend or used APIs. Properties can also represent states necessary for the correct *visual rendering* of the component, like the currently selected map type in Google Maps or the chosen theme in a text editor. Finally, properties can represent parts of the underlying *data* model, which can be both represented directly or indirectly on the UI.

As indicated by the highlighted fields in Fig. 1, our work uses seven of the analyzed elements to differentiate six primary coupling levels presented on the right. To support non-programmers best, the number of levels should be kept at a minimum. To consider the most significant cases of typical collaborative scenarios only, synchronizing options like recommendations or the adaptivity behavior are intentionally excluded. Coupling levels are organized as a hierarchy from top to bottom. Tighter, upper levels subsume more loose, lower ones. Likewise, arbitrary level combinations are not intended. The assumption, that a hierarchy fosters simplicity and clarity of the concept for daily Web users with no programming skills was proven by our user study (cf. Sect. 5). The definition of tightness rest on the number of access options coupled, which is used as the primary order criterion. The tightest level is denoted with *screen flow + viewport*. At this level, all parts of a CWA are synchronized including transitions between screens or scroll and zoom events at the viewport. The layout is the same for all users, which can cause problems in case of different screen sizes or resolutions.

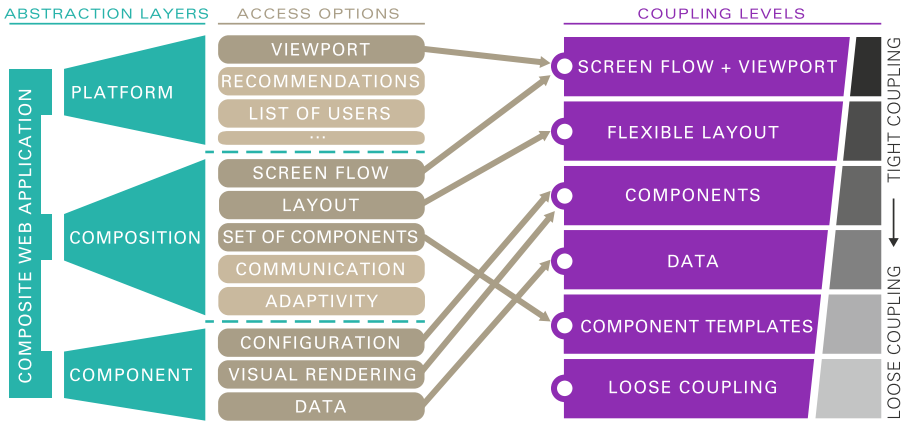

Fig. 1. Derivation of possible coupling levels for composite web applications

The *flexible layout* only synchronizes layout information about the components order, relative size of the viewport, and position within the grid system. In contrast to the tighter level, users will receive an optimal user experience for devices with different screen sizes or resolutions while still be able to work tightly coupled. The level *components* allow for an individual layout configuration of each participant. While components' order and size as well as screen flow and scroll events are not synchronized, each component of the application, including its configuration, visual rendering, and data model are synchronized. To support cases, where participants need individually configured component UIs, like for example individual scroll positions in a text editor or different map types on a map, the level *data* can be used. Within each component, only the properties representing the data model are synchronized. To support use cases, which are more based on coordination than collaboration, the last two levels can be used. *Component templates* only couple the initial set of components shared by the owner as templates for initializing the applications of all participants. Afterward, all group members work individually with the same set of components and can synchronize only single data elements like the current list of hotels or a set of locations. *Loose coupling* indicates a level, where every collaboration partner can individually choose components to work with. Again, the only synchronized connection between each user can be represented by a single data element, which represents the task to be accomplished.

3.2 Sharing Modes

Since the differentiation of the presented coupling levels is hard for end users with no programming experiences, we additionally introduce seven sharing modes. These sharing modes provide an end-user-oriented abstraction layer by subsuming coupling levels with suitable sharing objects and permissions. Thereby, each mode

represents a typical collaboration scenario by a suitable identifier, easy to understand for non-programmers. The following seven modes can be distinguished:

Tight presentation mode represents the most restrictive mode by adopting the level of *screen flow and viewport*. It enables an inviter to present content without interruption. This mode automatically shares the entire application with the permission only to view state changes (see first sharing of Bob within the reference scenario).

Free presentation mode utilizes the level of *flexible layout* to support participants with different devices and screens while enabling the inviter to present content tightly coupled. The entire application is shared under view permission by default.

Tight collaboration mode represents the combination of the coupling level *components* with the permission to edit the content of the shared parts. This enables a collaborative usage of the entire application or single selected components. Thereby, all shared components have identical and synchronized visual appearances.

Free collaboration mode similarly is based on the permission to edit but uses the coupling level *data*. This implies that users can work synchronously on data with an individually configured visual appearance of the shared components.

Tight individual mode targets asynchronous scenarios where collaborative partners use a fixed set of components, which is not synchronized. This mode grounds on the level *set of components*. By selecting the permission *edit* for single component properties, for example, the inviter can force participants to independently search for hotels and exclusively synchronize their result list with the inviter.

Free individual mode in contrast to the previous mode, participants can choose alternative components from a set of given components. This can be used to define a clear workspace for the invitees, but consider their personal preferences, for example by using Bing as map provider instead of Google.

Coordination mode represents the level of *loose coupling*. Within this mode, invitees are completely free in their choice of the right components for the assigned task of the inviter. By assigning edit permissions to single properties, the inviter can create a set of tasks in form of desired data elements to be filled by his collaborative participants. The platform recommends suitable components for the invitees.

Even if some modes includes predefined, not changeable sharing objects, it is possible to customize single component properties. For each mode, the inviter can define individual data elements to be private or only viewable instead of editable. How this can be done, is explained in the next subsection.

3.3 Coupling-Aware Sharing Process

Figure 2 presents the process for creating sharing definitions for CWAs, which extends the initially introduced process of [6]. The extended process comprises

three steps to invite new users or extend existing collaborative sessions. In*step 1*, one of the seven sharing modes introduced in the previous section has to be selected. Selecting the sharing mode, and therefore the desired coupling level has do be done first because it implies restrictions for the set of selectable objects and permissions. *Step 2* includes the definition of the sharing triple. As a subject, arbitrary combinations of groups and single users can be defined ①. For each subject, tuples of object and permission can be attached ②. Thereby, users can select objects and permissions which are allowed for the current sharing mode. In general, possible objects can be arbitrary composition fragments (entire application or single components), composition features, or composition data ③. Permissions at this step can be *edit* or *view*. With the second step, multiple triples of subject, object, and permission can be defined ④. The final *step 3*, optionally allow for fine-grained customization of permissions for single fragments in form of composition features or data ⑤. This includes, for example, the selection of confidential data to be hidden as private for other participants. Thereby, the concepts presented in [7] are being applied. Independent of the current sharing mode, single fragments of the selected sharing object can be selected to customize their specific permission. Users can assign view or edit permissions to single fragments. This can be used especially in the case of coordination, where participants use, for example, the *free individual mode*. Because this implies no general synchronization, single fragments can be specified as to be synchronized for exchanging, e.g., the result of a hotel search. If the object equals the entire application, then single components, features or data elements can be selected. If single components were selected initially, only features and data can be chosen here. Within each sharing definition, steps 2 and 3 can be iteratively repeated ⑥. Finally, the creation of a new sharing definition is finished by sending the invitation to join a new or extend an existing collaborative session to the users specified as subject. Afterward, anytime during the collaborative session, users can start creating or adopting other sharing definitions ⑦.

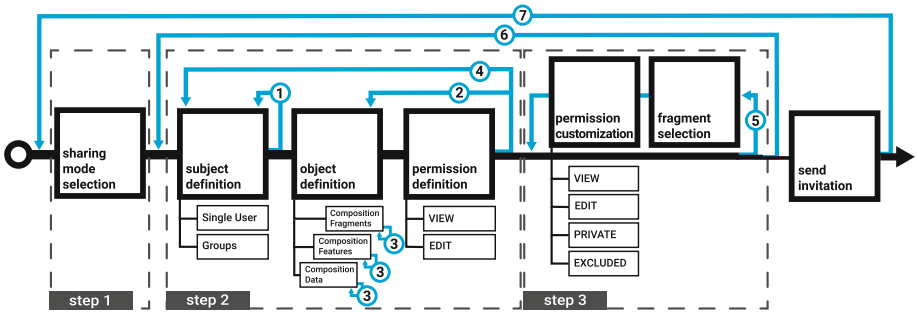

Fig. 2. Process of creating a sharing definition

4 User Interface and Awareness

The defined sharing process comes with integrated visual tooling support, which is based on the UI concept presented in [6]. This section describes the extensions made to consider different coupling levels during sharing and the awareness features in live view to review applied settings.

4.1 Defining New Sharings During Runtime

Step 1, 2, and 3 of Fig. 2 form the basis for a wizard-like interface concept. Figure 3 includes four screenshots which exemplify their practical execution.

As indicated in (A), the process of creating a sharing definition starts by opening a modal dialog. After a short introduction text, as defined in the sharing process, users have to select their desired sharing mode first. To ease the understanding of the seven sharing modes introduced in Subsect. 3.2 for non-programmers, an additional level of abstraction was introduced. By following typical practical, collaborative activities, users first have to select the intended purpose of their collaborative session in form of *present* some information, *work together* with others, or *assign tasks*. This lowers the initial set of possible choices to three instead of seven and therefore, reduce the users' cognitive load. Screen (A) highlights the case, when a user hovers the *work together* option. After selecting this purpose, (B) indicates, that only suitable sharing modes are visualized. In this case, the user first hovers, later selects the *free collaboration mode*. *Present* covers *tight* and *free presentation mode*. *Tight* and *Free Individual Mode* as well as *Coordination Mode* are represented by *assign tasks*. Within each mode, a short description as well as a bullet list of synchronized elements are displayed, to ease their differentiation for non-programmers. The *user guide* on the right permanently displays additional information depending on the current step and its current configuration.

After finishing step 1, as indicated in (C) of Fig. 3, users are asked to configure their basic sharing definition by visually composing a triple of *who*, *what*, and *how*. According to the selected sharing mode, *what* and *how* may be preselected and can not be changed (cf. Subsect. 3.2). In any case, the user has to specify a set of users or groups which should receive an invitation for sharing. Here, Alice was selected as subject for the triple. Details about the definition of triples were already presented and evaluated by [6] and therefore, are not in scope of this paper.

Within the final step (D), users can customize permissions or hide private data. This step is optionally and can be skipped by directly clicking at the *share* button, which was activated by opening this step. The *preview* button can be used, to review the current settings before sharing them by using an impression view introduced by [7]. Step 3 further extends the results of [7], which introduced an UI for visually selecting data elements and component features to be hidden as private within a collaborative session. The screen lists all data elements shared, grouped by the components they are originating from. This applies for the components' features too. Because the user has chosen the

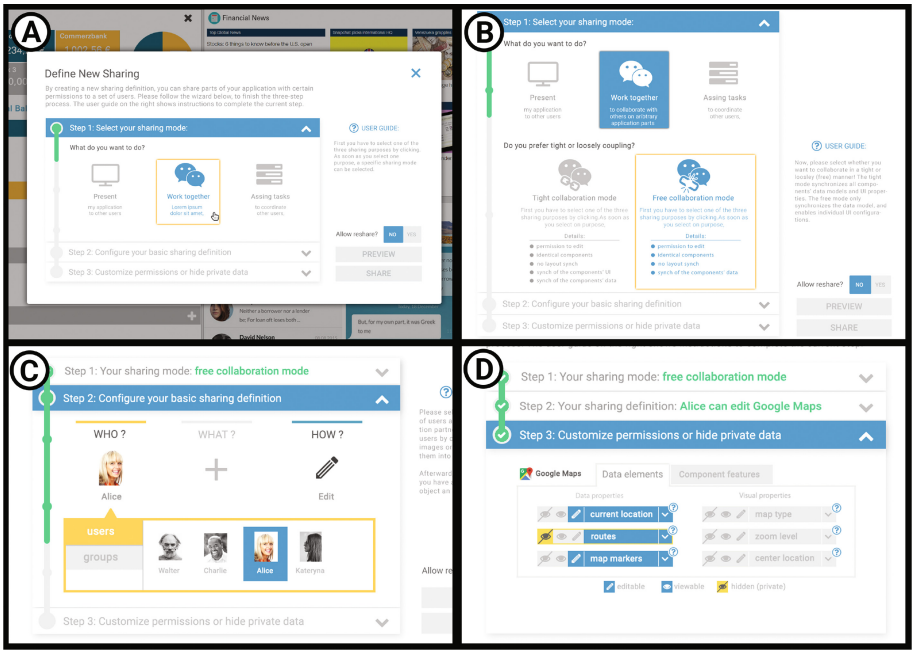

Fig. 3. UI support for new sharing definitions with multiple coupling levels

free collaboration mode in this example, all component properties configuring the visual appearance are skipped from synchronizing and therefore visually grayed out. For this sharing mode, all data elements are marked as synchronized per default indicated by their blue color. In addition, each element receives the basic permission specified in step 2, in this case *edit*. The user can now change the permission of each data element to either *viewable*, *editable*, or *private*. The last implies, that the element is handled as private following the concept of [7].

In case one of the three coordination modes were selected (purpose *assign tasks*), no data element or component feature gets synchronized and therefore is marked as active within this panel. Thereby, single elements can be activated, to indicate, that the list of hotels or the current location should be synchronized as a result of the assigned task. The whole process is assisted by a green status indicator at the left. It displays a green check mark for each successfully processed step. In addition, each step includes small circles indicating the number of actions necessary within.

After finishing a sharing definition, the platform notifies the selected participants and changes to the application's live view. After an invitee joins, the platform ensures the correct synchronization of corresponding application events, like scrolling or adding components, and component state changes, like a changed data property, depending on the conceptual specifications of the selected sharing mode (c.f. Sect. 3).

4.2 Live View Awareness and Adaptation

After creating some sharing definitions for different collaborative partners, it is necessary to constantly be aware of all granted permissions and coupling levels. As indicated in Fig. 4, for each shared component of an application, a small pop over menu can be opened by clicking on the icon that indicates the number of collaborating users for this component. The dialog is divided into an overview at the right side (E) and a detail view for one selected participant at the left (F).

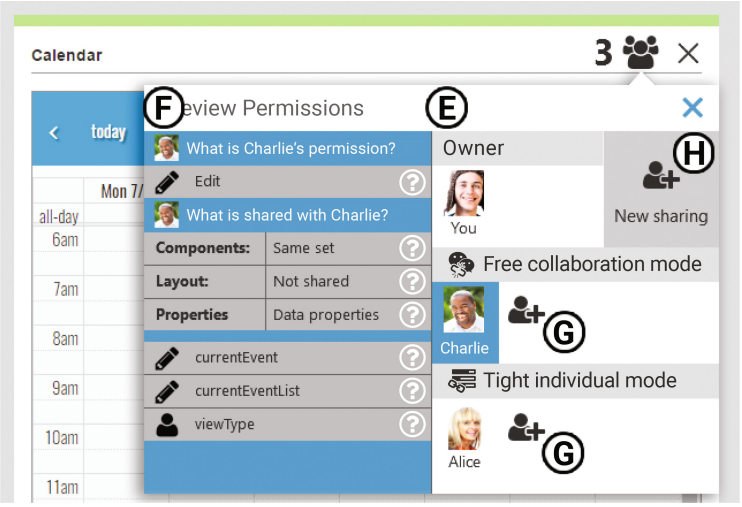

Fig. 4. Awareness for current sharing definitions

The overview visually separates all participants by their assigned sharing mode. By clicking at the plus icon (G), a new sharing definition with the same sharing mode can be created. Therefore, the sharing dialog of Fig. 3 opens by initially showing step 2 (C). To start a new sharing definition with a different sharing mode, the *new sharing* icon at the upper right can be used (H). The current screen shows the perspective for Bob, the owner of the component, who has invited Charlie and Alice. If Alice or Charlie would open this view, they only can check their own sharing mode and the name of the owner.

The detail view initially shows the active sharing mode and insights on the permissions for each data element for the current user. As soon as one user present at the overview gets hovered, the detail view changes and visualizes the permissions for them.

5 Evaluation

The concept of different coupling levels for collaboratively used, composite Web applications is evaluated with regard to its practicability by a reference implementation within the Composition of Rich User Interface Services for Everybody

(CRUISE) runtime environment (Subsect. 5.1). To prove the user acceptance, a user study was conducted (Subsects. 5.2 and 5.3).

5.1 Implementation

The CRUISE platform hosts mashup applications to enable their collaborative usage and reconfiguration during runtime. Therefore, a server-side coordination layer is implemented as a singleton using Enterprise Java Beans (EJBs) for all clients. The client side is realized by standard, modern Web technologies (HTML5, CSS and JavaScript). Client-server communication is done by Web sockets using Apache Apollo[4] The representation of the coupling levels is realized as an extension of the access control list (ACL) implementation already presented. The access control list (ACL) on the client side is represented via JSON and on the server side via Java. After creating a new sharing definition or changing the coupling level during runtime, dedicated commands are used to synchronize updates between client and server to keep the ACL consistent for all clients. To realize the UI support, we mainly used Bootstrap[5] and Google Material Design in form of the front-end framework propeller[6]. This ensures fast development, responsive design, and consistent user experience.

5.2 Methodology

A preliminary user study was performed to evaluate user acceptance and experience. As the methodology for our experiment, we used the think-aloud protocol. Participants had to pass a tripartite setup. First, a general introduction on how to create, use, and modify composite Web mashups with the EUD-mashup platform of CRUISE as well as on the reference scenario presented in Subsect. 1.2 was given. Second, participants were asked to complete four tasks with increasing complexity by using an interactive click prototype, that was created with the tool *invision*[7]:

- Share the calendar component and select an appropriate coupling level which allows others to choose an individual calendar formatting style.
- Share the map component to present your marked locations, consider individual screen resolutions, and ensure, that invitees can only change the map markers.
- Review and explain the present awareness information, in case of a not valid and a not meaningful sharing definition (created by the moderator).
- Log in as an invitee. Explain the sharing awareness features by exemplifying configured coupling levels and permission restrictions.

[4] https://activemq.apache.org/apollo/.
[5] http://getbootstrap.com/.
[6] http://propeller.in/.
[7] https://www.invisionapp.com/.

A moderator noted comments. Finally, the participants were asked to fill in the System Usability Scale (SUS) questionnaire to get a comparable, standardized result.

As presented in Table 1, the study was conducted with the help of six male and three female participants including ages from 21 to 48 years (29 on average). All daily use the Web. Seven had no or basic programming skills. They fit the target group best. Professions were widely spread, including physics, economy, engineering, accounting, earth science, and architecture.

5.3 Results

The overall feedback from all participants was very positive. Selecting the appropriate sharing modes was in general no problem for the participants. Since the initial version of the UI showed all seven sharing modes in parallel, three users reported being overwhelmed. As a consequence, we introduced the purpose selection of step one (see ① in Fig. 3). Understanding the purpose of each sharing mode was easy for almost everybody due to the used icons and short descriptions. However, five participants reported, that they had to read the description of all modes to be sure about their selection at the initial test. In the original version, five users directly tried to click on the share button after selecting the sharing mode. To emphasize the necessary inputs in step two even more, the progress bar on the left side and some additional animation for the transition to step two were introduced.

Within step two, as proven in previous tests, users confirmed the simplicity of creating a sharing triple again. Similarly, the preselection of a permission originating from the selected sharing mode was clear for all participants. Six users had problems to understood that the whole application was preselected as the object. We added a visual and textual hint on that.

Within step three, six participants criticized the icon for private and not synchronized data elements, which we adjusted afterward. Visual hints for incorrect or missing definitions, tested by task three, were understood by everybody. The awareness features were intuitively opened and interpreted correctly by seven participants. Two had problems to identify that the icon at the components top bar can be clicked. Overall, the visual separation within the awareness view was rated as very positive. The only big misunderstanding within this view originated when users clicked on the provided help icons. The initial version presented a textual description of the sharing mode. Since users expected a live preview of the given permissions, the impression view, which was introduced in [7] is reused here.

In general, users were very interested in the idea of the overall platform and confirmed that they could imagine to use it for their daily activities. Once again, we observed, that self-descriptiveness is a crucial requirement for Web systems today. Participants only rarely read hints and use help icons. They prefer the paradigm *exploration before reading*. All participants filled in the SUS questionnaire and created an average score of 81.67, which indicates a good value for the general usability of the evaluated concepts. Individual results are listed in Table 1.

Table 1. Characteristics and SUS ratings of study participants

Age	Sex	Profession	Programming skills	Web usage	SUS rating
21	♂	Business informatics student	Advanced	Daily	92.5
24	♀	Business student	Beginner	Daily	95
26	♂	Business engineering student	Beginner	Daily	85
23	♂	Mechanical engineering student	Advanced	Daily	80
23	♂	Physics student	Beginner	Daily	75
48	♂	Food economics	None	Daily	77.5
48	♀	Accounting	None	Daily	72.5
25	♀	Landscape architecture student	None	Daily	80
24	♂	Earth sciences student	Beginner	Daily	77.5

6 Related Work

Today, only few a platforms allow for collaborative usage of EUD mashup applications. Approaches like PEUDOM [8], MultiMasher [9], or the vision of Tschudnowsky et al. [10] more or less enables users, to define different permissions for components of their composite application for synchronized usage. Nevertheless, none of the existing solutions discusses the provision or change of coupling levels at all. Sharing is based on a fixed, implicit degree of coupling whereby users never can review details about it.

Cooperative learning environments like [11] or [12] allow sharing parts of their workspaces with a set of users while keeping other parts as private. In difference to mashup platforms, they focus files or texts as content only and do not support interactive content. Although visual permission management metaphors exist, defining different coupling levels for various work scenarios is not part of these solutions.

Dewan and Choudhary [3] analyzed different, flexible coupling modes. Thereby, Boolean attributes indicate for each data property of an application whether value, format, or view coupling is desired. It is not possible to differentiate the coupling mode for different users. Also, users have to decide on their own about the meaningfulness of a configuration, which excludes end users without programming skills.

Gutwin and Greenberg [5] coined the phrase mixed-focus collaboration and discussed necessary awareness in these situations. They analyzed workspace navigation, interaction with artifacts, and view representation, which served as the foundation for this work. However, the analysis is very generic in the scope of this work. No runtime support for sharing parts of an application with arbitrary coupling levels is provided. Pinelle and Gutwin [13] separated loosely and tight coupling by interdependence, differentiation, and integration. Although the presented *Mohoc* framework required the mechanisms to change between these levels, more fine-grained level definitions and the adoption for non-programmers and their need for runtime changes were not covered.

The Component Groupware of ter Hofte [14] provides means for addressing different coupling levels. Similar to our classification, they define a hierarchy of four status levels which result in four coupling levels. Concerning the research challenges of this work, this classification was used as a foundation, but is too broad and offers no runtime support for our target group. Inter alia, the work was extended by [15] by providing components exchangeable during runtime. Within a similar approach, the usage of the proposed components enables to switch the coupling level during runtime between interface, user, collaboration, and resource. However, also here detailed concepts for supporting users to evaluate and configure the right coupling level are missing.

Tandler [4] provided an environment for ubiquitous computing which considered coupling levels based on data, application, user interface, environment, and interaction model. As before, the results have to be detailed to work for composite Web applications and provides no runtime support for finding and reviewing required coupling levels.

Overall, none of the approaches enables non-programmers to define or adjust multiple coupling levels for their shared composite Web applications. Despite, many classifications and aspects of elements relevant for coupling exist, none sufficiently covers the particular challenges of CWA and none provides runtime support for non-programmers.

7 Conclusion and Future Work

By facilitating the mashup paradigm and universal composition, situational Web applications were proven to be intuitive and easy to use for collaborative scenarios. Even users with no programming skills are empowered to do user-driven development by the provided features for individual application tailoring during runtime. However, the analysis of related work showed significant deficiencies when considering different coupling levels for collaborative Web applications based on black-box components. This includes missing classifications and useful abstraction layers for coupling levels as well as UI support to enable the target group of users to share applications on their own.

This paper has three contributions. It presents an analysis and classification for elements in composite Web applications relevant for coupling. Besides, the clustering of theses elements combined with preselected permissions and sharing objects results in seven common sharing modes, which can be used by non-programmers to fulfill individual collaboration needs efficiently. To this end, any user is enabled to manage multiple coupling levels based on an integrated tooling during the initial specification of new sharing definitions as well as the review and awareness of these during runtime. The suitability of this approach for users with no programming skills is approved by a user study including nine 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 81.67, which we found to be a good outcome. The feedback from the user study is already considered within the improved concept discussed in this paper.

Future work includes the adjustment of the existing workspace awareness approach. Currently, users can integrate proper awareness information as widgets on their own. It is not possible yet, to filter the set of available widgets, e.g., based on rules like presented in [5]. The goal is an excellent alignment of coupling level and selected awareness widgets. Nevertheless, we believe, that the proposed work is an essential factor for the acceptance of Web-based collaborative platforms in real life scenarios. Due to their generic nature, it is essential to support arbitrary application and collaboration scenarios by the coupling levels needed from daily Web users to fulfill their tasks successfully.

Acknowledgments. The work of Gregor Blichmann is funded by the European Regional Development Fund and the Free State of Saxony.

References

1. Pietschmann, S.: A model-driven development process and runtime platform for adaptive composite web applications. Technology **2**(4), 277–288 (2009)
2. Daniel, F., Casati, F., Benatallah, B., Shan, M.-C.: Hosted universal composition: models, languages and infrastructure in mashArt. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 428–443. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04840-1_32
3. Dewan, P., Choudhard, R.: Flexible user interface coupling in a collaborative system. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 1991, New Orleans, Louisiana, USA, pp. 41–48. ACM (1991). doi:10.1145/108844.108851. ISBN: 0-89791-383-3
4. Tandler, P.: Synchronous collaboration in ubiquitous computing environments: conceptual model and software infrastructure for roomware components. Ph.D. thesis. Darmstadt University of Technology, Germany (2004)
5. Gutwin, C., Greenberg, S.: Design for individuals, design for groups: tradeoffs between power and workspace awareness. In: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, CSCW 1998, Seattle, Washington, USA, pp. 207–216. ACM (1998). doi:10.1145/289444.289495. ISBN: 1-58113-009-0
6. Blichmann, G., et al.: 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, Rome, Italy, 23–25 April 2016, vol. 2, pp. 17–26. SciTePress (2016). doi:10.5220/0005862800170026. ISBN: 978-989-758-186-1
7. Blichmann, G., et al.: Private data in collaborative web mashups. In: Proceedings of the 13th International Conference on Web Information Systems and Technologies, WEBIST 2017, Porto, Portugal, 25–27 April 2017 (2017, in press)
8. Picozzi, M.: End-user development of mashups: models, composition paradigms and tools. Ph.D. thesis. Politecnico di Milano, March 2014
9. Husmann, M., Nebeling, M., Norrie, M.C.: MultiMasher: a visual tool for multi-device mashups. In: Sheng, Q.Z., Kjeldskov, J. (eds.) ICWE 2013. LNCS, vol. 8295, pp. 27–38. Springer, Cham (2013). doi:10.1007/978-3-319-04244-2_4

10. Tschudnowsky, A., et al.: 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, pp. 193–200. SciTePress (2014). doi:10.5220/0005049001930200. ISBN: 978-989-758-043-7
11. Schümmer, T., Haake, J.M., Haake, A.: A metaphor and user interface for managing access permissions in shared workspace systems. In: Hemmje, M., Niederée, C., Risse, T. (eds.) From Integrated Publication and Information Systems to Information and Knowledge Environments. LNCS, vol. 3379, pp. 251–260. Springer, Heidelberg (2005). doi:10.1007/978-3-540-31842-2_25
12. Bogdanov, E.: Widgets and spaces: personal & contextual portability and plasticity with opensocial. Ph.D. thesis. Ecole Polytechnique Federale de Lausanne (EPFL), August 2013
13. Pinelle, D., Gutwin, C.: A groupware design framework for loosely coupled workgroups. In: Gellersen, H., Schmidt, K., Beaudouin-Lafon, M., Mackay, W. (eds.) ECSCW 2005. Springer, Dordrecht (2005). doi:10.1007/1-4020-4023-7_4. ISBN: 1-4020-4022-9
14. Henri ter Hofte, G.: Working apart together -foundations for component groupware. Ph.D. thesis. Telematica Instituut, Enschede, the Netherlands (1998). ISBN: 90-75176-14-7
15. Farias, C.R.G., Gonçalves, C.E., Rosatelli, M.C., Pires, L.F., Sinderen, M.: An architectural model for component groupware. In: Fukś, H., Lukosch, S., Salgado, A.C. (eds.) CRIWG 2005. LNCS, vol. 3706, pp. 105–120. Springer, Heidelberg (2005). doi:10.1007/11560296_8

http://www.springer.com/978-3-319-63873-7

Collaboration and Technology
23rd International Conference, CRIWG 2017,
Saskatoon, SK, Canada, August 9-11, 2017,
Proceedings
Gutwin, C.; Ochoa, S.F.; Vassileva, J.; Inoue, T. (Eds.)
2017, X, 261 p. 66 illus., Softcover
ISBN: 978-3-319-63873-7