Seamless Application Ecologies as Mobile Personal Learning Environments

Christoph Greven[1], Navid Gooranourimi[2], Shima Amin Sharifi[3], Hendrik Thüs[4], Mohamed Amine Chatti[5] und Ulrik Schroeder[6]

Abstract: The demands on today's learner have changed over the recent years as the understanding of his role evolved. Getting away from the traditional top-down approaches, the learner is more and more in control of selecting learning resources or choosing the right learning techniques. On the one hand, by following the idea of a learner-centric Personal Learning Environment it is now possible to freely design an own learning environment with all required tools to improve the learning process. On the other hand, the ubiquitous usage of mobile devices such as smart phones or tablet computers creates the basis for new modern learning scenarios. While different applications for the devices already allow an individual aggregation of tools they are still highly separated and cannot interact beyond the borders of one application. In this paper, we present a new approach realizing application ecologies on mobile devices which allow integrating services and functionalities of one application into another one. This may happen ad-hoc without involvement of application developers resulting in high extensibility. Furthermore, it allows the user to adjust the app system according to his needs and creates the impression of one seamless mobile PLE consisting of an adjustable set of applications.

Keywords: seamless, application ecology, mobile, PLE, intercommunication, integration, borderless

1 Introduction

The understanding of learning by itself has evolved over the recent years implicating the demands on learners and their learning tools. It has been recognized that traditional formats alone cannot cope with the tremendously fast moving information and knowledge. That is not only restricted to the sector of public education and private life but also to professional contexts like companies which comprehend knowledge as the competitive

[1] RWTH Aachen University, Learning Technologies Research Group, Ahornstraße 55, 52074 Aachen, greven@cs.rwth.aachen
[2] RWTH Aachen University, Learning Technologies Research Group, Ahornstraße 55, 52074 Aachen, navid.gooranourimi@rwth.aachen
[3] RWTH Aachen University, Learning Technologies Research Group, Ahornstraße 55, 52074 Aachen, shima.sharifi@rwth.aachen
[4] RWTH Aachen University, Learning Technologies Research Group, Ahornstraße 55, 52074 Aachen, thues@cs.rwth.aachen
[5] RWTH Aachen University, Learning Technologies Research Group, Ahornstraße 55, 52074 Aachen, chatti@cs.rwth.aachen
[6] RWTH Aachen University, Learning Technologies Research Group, Ahornstraße 55, 52074 Aachen, schroeder@cs.rwth.aachen

advantage. Hence, tools are created to support the new learner in his self-organized learning process with a bunch of functionalities to react situationally to occurring problems. With the potential of mobile computing learning environments are also shifting to mobile devices like tablets. Because so far there are only independent autarkic applications, navigation is difficult and context information gets lost when switching between different tools.

In this paper we present a seamless application ecology approach where mobile applications can automatically share their features beyond the borders of one application to suggest one single seamless mobile personal learning environment that can be adapted to the individual needs of the learner.

The remainder of this paper is structured as follows. Section 2 gives an overview of the concept of Personal Learning Environments, some implementations and the occurring problems on current mobile systems. A new approach to solve those problems is presented in section 3 while section 4 shows a concrete realization for Android devices. Section 5 finally gives a conclusion followed by an outlook of possible future work.

2 Personal Learning Environments

In this chapter the concept of Personal Learning Environments (PLE) is introduced and some popular representatives are presented. Furthermore, the current problems with PLEs in form of mobile applications are discussed.

2.1 Concept of PLEs

The term Personal Learning Environment (PLE) first came up in the beginning of the 21st century when discussions arose whether traditional, formal teaching formats and their technical implementation such as Virtual Learning Environments are able to cope with the evolution of learning. There are various different definitions what Personal Learning Environment actually means, but beyond all it can be seen as a general concept that puts the learner in the center of the learning process empowering him to individually shape the own learning environment [Ca10]. While, in principle, the whole learning environment, e.g. the whole office, can be called a PLE, literature tends to concentrate on technical implementations that support the aggregation of lightweight and loosely coupled tools and services that support the learning process. Besides the self-regulated learning, the aspect of knowledge reflection, exchange and networking is fundamental for a PLE. This is also stressed in the LaaN Theory of Chatti et al. [CSJ12] which understands the learning process as the continuous creation of the Personal Knowledge Network. According to Attwell et al. [At08] a Personal Learning Environment should support in the following activities: Aggregate and scaffold, manipulate, analyze, reflect, present, share, and network. That clearly goes beyond the scope of actual Learning Management Systems which are more content-centric, closed, centralized and commonly

controlled by one authority or organization to support formal learning in a structured way top-down. In contrast, a Personal Learning Environment is highly personalized and emergent, and is also able to reach informal and life-long learning, not being restricted to formal factors like the time span of a course. The following section shows some exemplary frameworks that can be categorized as PLEs.

2.2 PLE Realizations

Personal Learning Environments or rather the tools and frameworks which enable the learners to create one are the topic of this section. There is neither a general or common architecture for a PLE nor a right or wrong way to realize it. However, most tools can be understood as a kind of collection and combination of services. This concept, namely mashup, can be divided into two main groups: mashups by aggregation, and mashups by integration [Ca11]. Mashups by aggregation more or less simply aggregate different external services and tools into one surrounding frame. The individual parts are highly independent and can be combined ad-hoc without further engagement. Starting with the desktop of a personal computer they have evolved over time with stronger technologies. As an example, the personalized homepages (PSP) like the former iGoogle, Netvibes[7], or Pageflakes[8] might be mentioned. Mashups by integration include more intelligence and require communication between different components. Results delivered by one fragment can be used as input for another one. This way, many small and simple tools are nested to a more powerful instrument which is able to address a concrete purpose. While in the beginning the interconnection required some programming skills to be set up via formal languages, newer approaches enable users without previous knowledge to easily integrate their ideas. Popular examples are Moodle[9] or ROLE[10]. They are taking this to extremes so that users can even create their very own small gadgets or plugins like described by Soylu et al. [So11].

The idea of atomic widgets, gadgets, tools, or applications catches up the concept of PLEs very well. The concept has been applied on desktop computers since long and renewed by Microsoft Window's tiles. It has been used as web applications in recent years and has also been transferred to nowadays' mobile devices such as smartphones and tablet computers. The PRiME[11] project [Gr14] is a good example how this could be applied. While mobile technology offers many new possibilities there are also some problems occurring which are substantiated in the next section.

[7] http://www.netvibes.com
[8] http://www.pageflakes.com/
[9] http://moodle.de/
[10] http://www.role-widgetstore.eu/
[11] Professional Reflective Mobile Personal Learning Environments; http://prime.rwth-aachen.de

2.3 Drawbacks of Mobile Apps as PLEs

The concept of applications on mobile devices such as smartphones and tablet computers heavily supports the idea of a Personal Learning Environment. The user can install an individual set of applications according to the current needs which can cover various types of tasks. In case of the Android operating system and the related application stores such as the Google Play Store, apps can even be created and published by users themselves. Although programming skills are needed, the choice of applications is enormous [Nu15]. As single applications normally only cover one stand-alone functionality, one problem arises: Apps are closed, independent and do not support interoperability. As a result, the pivot of all navigation is the device's home screen. All features can only be accessed from there by choosing the corresponding right application first. The respective context of the user in which the need for a certain tool arose gets lost and further information from that situation cannot be transferred to the required service. This independence of applications is not only recognizable in the different functionalities but also clearly reflected in the visualization of the graphical user interface. The appearances differ a lot in regard to color, style, composition, navigation, etc.

Nevertheless, in principle a joint application ecology with interrelated applications could be realized. In this case, the contract points of all involved applications need to be set up in advance in the development phase meaning that the developer is engaged in the process. Due to the open market, the theoretically unlimited amount of applications and their constant evolution, this approach cannot work in reality. If there was a new application available or changes were made, all other applications would need to be adapted to the newly available services. Id est one change of a service commonly results in a rework of the whole application ecology. This is not feasible and so far stand-alone applications have been created.

Obviously, with the current approach, a seamless application ecology with intercommunication is not realizable. The next chapter presents a concept to make this possible.

3 Concept of Seamless Application Ecologies

While the previous chapters introduced the topic Personal Learning Environments, presented its concepts and clearly discussed its problems especially in regard to mobile devices, this chapter presents a new conceptual approach to address this boundaries on mobile Android devices.

Commonly, mobile applications are realizing one basic functionality on a mobile device such as presenting content, sharing information, etc. Still, those functionalities may be important in the context of other applications. Hence, the main idea of this work is to connect the loosely coupled applications and offer their services and features beyond application borders. Doing so, applications can embed external functionality of other applications into their own environment and call the corresponding services respectively.

Softening the borders of the single application this leads to an overall seamless architecture where the user gets the impression of one single global and homogeneous learning environment. This interdigitating prevents duplicated and redundant functionalities throughout the whole system and creates an added value in different new contexts. While the developer/programmer of the application might define general conditions for the offered services or services to be used within one application, the user has the absolute control over his environment at runtime and can individually shape it according to the current needs. In contrast to recent approaches, applications installed at runtime can communicate their offerings ad-hoc. This means applications do not need to know about the whole application ecology beforehand and the developer is released from integrating known external features in the development process. Figure 1 shows an idealized interrelation of several applications. Application A and B offer services which are communicated to other applications such as application C. The user is then able to select from the offered set of applications and services respectively and embed it into another one. In the example, service A is included into application C meaning a graphical representation of it is added in the user interface, e.g. as a simple button. At the same time, the external service is integrated into the own application environment where it can be called and executed at runtime. Required context information can easily be added to the call such that the service is able to process it and adapt the behavior accordingly.

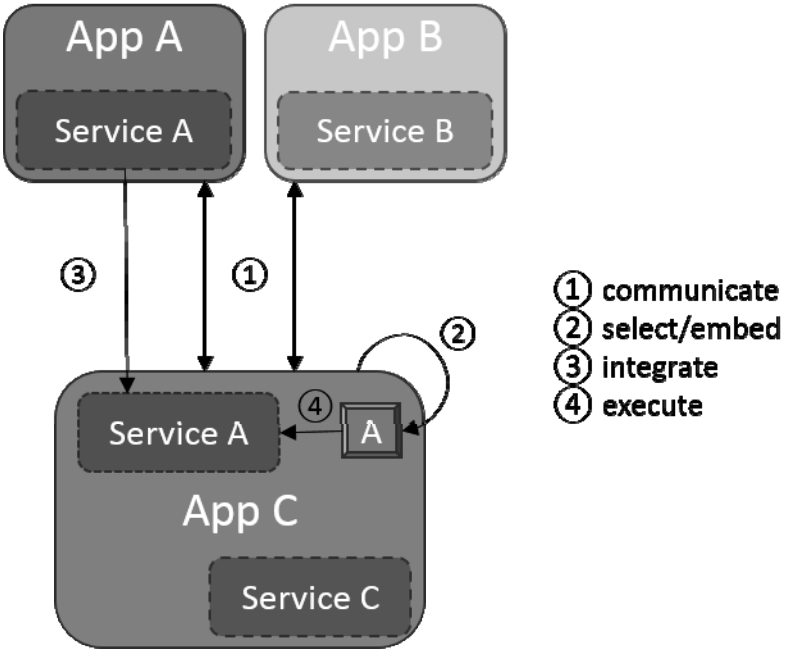

Fig. 1: Interrelation of two applications A and B where A embeds service S of application B

4 Realization

The implementation of the described mobile seamless application ecology concentrates on Android as an operating system due to its openness and its market share, e.g. more than 75% regarding smartphones in 2014 [So15]. It has emerged that this choice has some implications on the architectural design. In Android, applications consist of different activities[12] which represent basic functionalities of these applications. They are comparable to controllers that handle certain parts of the business logic. The concept of fragments[13] in Android further allows to portion the behavior and parts of the user interface in even smaller pieces and reuse those several times within one application. Upon now, there is no easy and comfortable way to allow direct reuse of fragments or activities (in regard to the physical execution environment) beyond the border of one application (see chapter Future Work for more details). Hence, there is no possibility to directly integrate features of applications into one another. It has been decided to realize a lightweight version through a central anchor application which manages all the service offerings. External functionality can be used and linked this way but is not physically included in the scope of a running application. Figure 2 shows how this manager is integrated into the scenery as a whole. New applications in the ecology can register their services actively at this central point. Afterwards, they are available for all other activities and can be embedded in their own user interfaces to make them accessible. In the following, the three components service, manager, and client are explained in more detail.

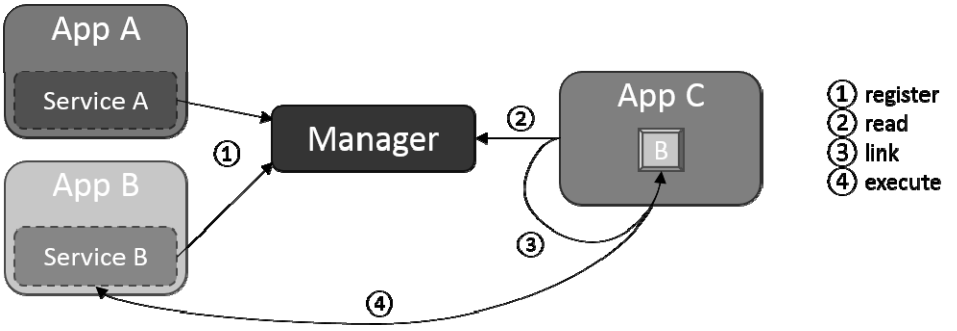

Fig. 2: Interconnection within the application ecology via a central service manager

As mentioned above, a *service* represents one feature or functionality of an application. Services are grouped by categories of activities they perform. This is similar to the different intent action types in Android (such as view, edit, or send), but allows more flexibility and more detail. The set of available activities is not enclosed and can be extended at any time. To clearly identify a service in the application ecology, the unique fully qualified class name of its activity is used. Installation of different applications with fully qualified class names is not allowed in Android out of the box. The goal of this work is a

[12] http://developer.android.com/guide/components/activities.html
[13] http://developer.android.com/guide/components/fragments.html

seamless embedment of one service into another application. Hence, some graphical elements are needed to realize the link in the graphical user interface of the client application. Thus, some image icons should be provided by the service which can be used elsewhere. They should have a high recognition value to ease usability and overview. Beside the technical implementation it is also very important to create the look and feel of one single application, although various different services from several applications are involved. Thereby, it is important to keep to certain common rules and create a visual seamless design. A collection of guidelines and best practices has also been developed in context of this work to support the design and development process. As stated earlier, services are able to respond to different contexts. More precise, a service is able to respond to different types of input and adapt the behavior accordingly. Those available input types need to be defined beforehand. For example, a multimedia editor could treat audio and video slightly different. Likewise, an output type can be defined for the service. It specifies which kind of data is produced and handed back to the calling instance. As an example, there are picker or selector services which allow the user to select an item which should be used in some context. E.g. selecting a user with whom some data is to be shared. After setting up these details, a service is ready to be used, but so far it is not accessible. Hence, when starting up for the very first time, it registers completely automatically at a central manager, which is described next.

To facilitate the communication process, a central *management* entity has been introduced. For services it offers the possibility to register. All the required information is stored in the local data base of the manager, viz. it holds a list of all available services and their options on the mobile device. These include fully qualified class name, service name, service category, app name, service icons, service version, input and output types, etc.. While the manager application administrates its data base, it also allows access by outer applications that use the same sharedUserId to gather the required information. To preserve a certain control, the access is restricted to a certain group of applications based on their signature. Only applications which have been signed by the same certificate as the manager application are granted access. In practice, they have been published by the same authority. For personal usage this might be of minor importance but for a large company those constraints are mandatory and are a comfortable way to define the scope of one application ecology. Now as the manager holds all needed data, client applications can request it following the same security issues as mentioned before. It is possible to preselect the list of returned services via filters on the activity category or the input or output type for example.

How the gathered information about the services is used is incumbent on the individual *client* application. But to keep the effort as minimal as possible, the developer or programmer of a new applications can rely on a delivered library that comes with two implementation concepts for the graphical user interface: the plugin menu, and the toolbar menu. Via both approaches, the developer is able to place a menu for the service offerings in the application by only adding two lines of code. They can be set anywhere on an

Android Menu[14]. Further adaptability is feasible and not restricted by this approach, e.g. that the user is able to drag and drop the menu somewhere it suffices his/her needs.

The *plugin menu* results in a single menu button (see figure 3) with which a user is able to open a plugin menu. According to the input/output types and categories defined beforehand, it shows different services available which are grouped by the different categories. Figure 3 shows a screenshot from an Android application where the plugin menu has been opened. According to the settings, there are 4 matching services arranged in 3 categories. By tapping on one service entry, the functionality is called respectively. Doing so, it hands over the current context of the client application. That is e.g. the currently selected element in the app like a selected word, a paragraph, or any other information. The service is then able to directly respond to this input and behave accordingly. Figure 4 visualizes how an external service is included in form of a pop-up menu. Technically the communication is actually based on Android's intent[15] principle. The plugin menu provides fast and easy access for the user while not occupying a large amount of space in the original view. It is consistent, and hence easy to find and offers a simple navigation basis for the users. There is one specialty regarding a plugin menu: As long as there is only one matching service to be shown in the menu (again through filtering by category and input/output or because no more applications holding services have been installed yet), the menu icon in the user interface is replaced by the icon of this service. Opening the menu is then no longer possible. Instead, pushing the button directly calls the service via one click. This concept keeps the user's effort as low as possible and avoids long navigation paths. Following this idea, developers can also integrate many different individual plugin buttons not only in the common menu bars but embedded throughout the whole application.

[14] http://developer.android.com/guide/topics/ui/menus.html
[15] http://developer.android.com/guide/components/intents-filters.html

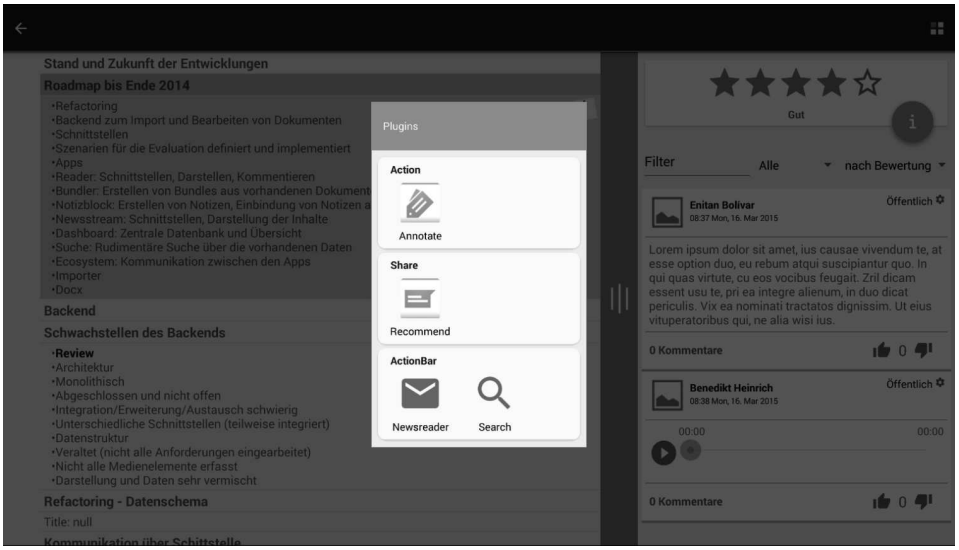

Fig. 3: A plugin menu offering 4 services in 3 different categories opened via a menu button in the top right corner

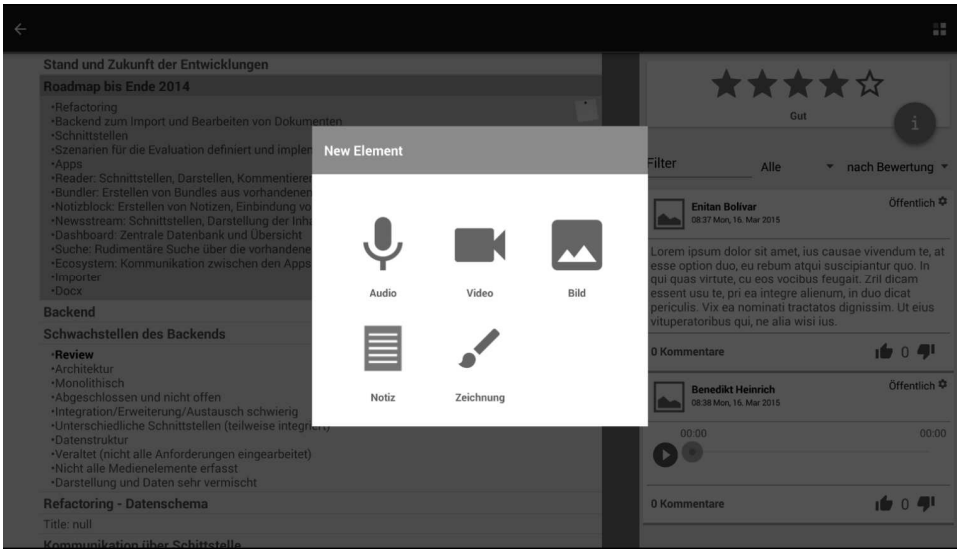

Fig. 4: An example of an external service opened as a pop-up window in another client application

The second approach is the *toolbar menu*. In contrast to the plugin menu, which is primarily under the control of the developer, the toolbar menu is designed to fit to the individual needs of the user/learner. It can also be placed on menus but does not only serve a

single one-fits-all button. Instead, it introduces a whole menu of tools. Its main button (the gear shown in figure 5) opens a popup, listing all available services again. The selection can be limited by the developer beforehand, so that the user is not overwhelmed by the choice. For each service the user can decide whether to display it in the menu bar (next to the main toolbar button) or not. Figure 6 shows a selection of two out of 4 available services. While changing the settings, the items on the corresponding menu bar are adapted accordingly. Beside the selection of services, the ordering of elements is also in control of the user. By simple drag and drop actions he/she can rearrange the items as indicated in figure 7. The ordering from top to bottom in the settings results in an ordering from left to right in a horizontal menu bar. The individual user adjustments are saved per toolbar menu while there might be an arbitrary number of toolbars in one application. The number of menus, their placements or the handling in general are detached from the concept of toolbar menus and can be either set by the developer or user at runtime. This way the learner is in control of his/her environment and can create and change it according to the current requirements. The developer fulfills the role of an advisor who can advise a set of elements in the menu, but no learner is bound to that preselection.

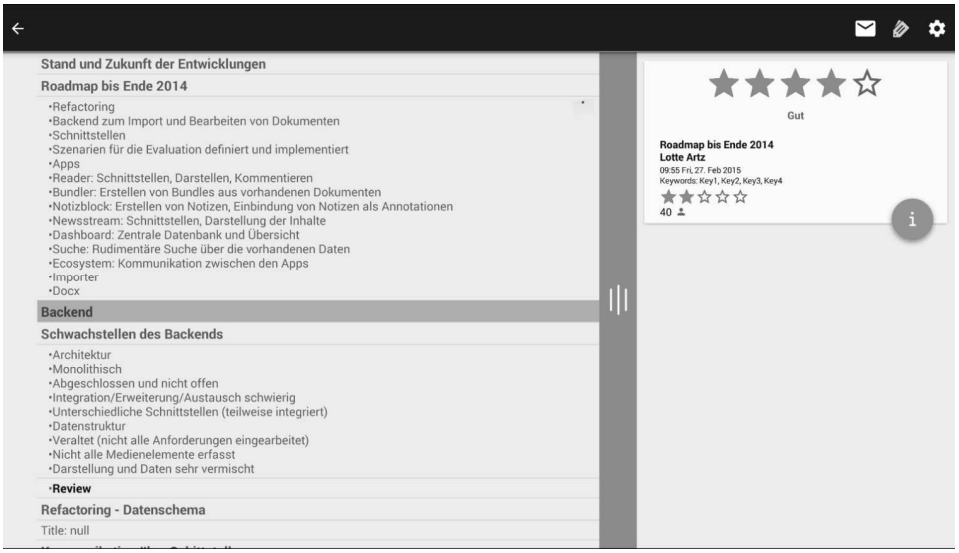

Fig. 5: An example toolbar menu (top right corner) with 2 services and the settings button according to the currently selected element

Summarizing, the two pre-implementations of plugin and toolbar menu offered for the developers keep the effort of extending an application to work in a seamless application ecology as low as possible. At the same time, the user receives tremendous possibilities in constructing his own learning environment.

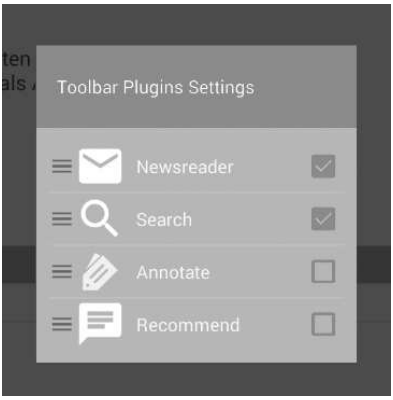

Fig. 6: Settings of a toolbar menu with 2 out of 4 selected services to be shown in the toolbar

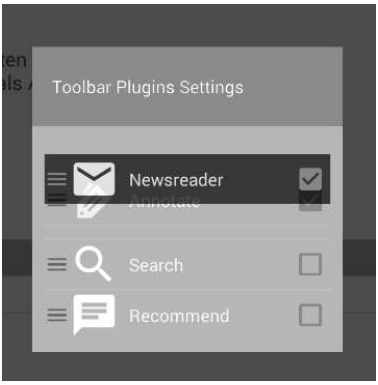

Fig. 7: Reordering of services for the toolbar menu

5 Conclusion and Future Work

5.1 Conclusion

In this paper we presented a new approach allowing the user to create a seamless application ecology as his/her individual personal learning environment. We therefore concentrated on mobile Android devices and highlighted a concept where applications are allowed to offer services to be integrated beyond the borders of one single application. Doing so, the learner can use external functionality by also taking into account the current situational context of the client application. To lower the implementation barrier for developers, we presented the two approaches plugin menu and toolbar menu which can be utilized in new applications via a library and few lines of code. Although the boundaries of usage can be set, e.g. by the different action categories of the services or their

different input types, the user is in control of his/her environment and can adapt the graphical user interface individually.

5.2 Future Work

As indicated in previous sections, Android does not allow to simply embed fragments of applications in one another. Reusing is only allowed within one single application. Therefore, we are currently only linking functionality but not really integrating it. A next step is to find a satisfying way to really embed parts of other applications. We could manage to include external fragments but that goes deep into how Java and its architecture works. Due to this rather hacky approach the performance is accordingly low. And, so far, there is only a uni- and not bidirectional way to include services. Once offering a service, a fragment can no longer include others services at the same time.

Currently, the central manager application acts like an anchor point and simplifies communication. Due to the context of the PRiME project, in which this approach has been realized, there is a central application anyway. Hence it was rather comfortable to use it for service management as well. In future and in general, this application in the middle should be bypassed. Direct communication between the different applications would keep the ecology more independent and flexible. New applications could either inform the existing environment about their functionality or embedding applications could ask the ecology for available services.

Due to the different settings a user can apply, he/she can already shape the environment to individual needs. Nevertheless, it would be interesting to see whether further adjustment possibilities would still support the users or be distracting already.

Recently, the concept could be evaluated in a preliminary inquiry with a small group of technicians in the context of the PRiME[16] project. According to the qualitative results of interviews, the presented approach seems to achieve a good overview, very fast access of tools, and personal adaptability. Inter alia, a larger and more detailed field study will be conducted in the second half of this year. Testers will then have the chance to intensively work with the system during their workaday life. Qualitative as well as quantitative results from interviews, think-aloud sessions and questionnaires will give more concrete feedback then.

References

[At08] Attwell, G.; Bimrose, J.; Brown, A., Barnes S.-A.: Maturing Learning: Mashup Personal Learning Environments. In (Wild, F.; Kalz, M.; Pakmér M.. Hrsg.): Proceedings of the First International Workshop on Mashup Personal Learning Environments, Maastricht 2008.

[16] Professional Reflective Mobile Personal Learning Environments; http://prime.rwth-aachen.de

[Ca10] Chatti, M. A.; Agustiawan, M. R.; Jarke, M.; Specht M.: Toward a Personal Learning Environment Framework. International Journal of Virtual and Personal Learning Environments, 1(4):71–82, 2010.

[Ca11] Chatti, M. A.; Jarke, M.; Specht, M.; Schroeder, U.: Model-driven mashup personal learning environments. International Journal of Technology Enhanced Learning, 3(1), 21–3, 2011.

[CSJ12] Chatti, M. A.; Schroeder, U.; Jarke, M.: LaaN: Convergence of Knowledge Management and Technology-Enhanced Learning. IEEE Transactions on Learning Technologies, 5(2):177–189, 2012.

[Gr14] Greven, C.; Chatti, M.A..; Thüs, H.; Schroeder, U.: Context-Aware Mobile Professional Learning in PRiME. In (Kalz, M.; Bayyurt, Y.; Specht, M. Hrsg.): Communications in Computer and Information Science. Springer International Publishing, 2014; p. 287-299.

[Nu15] Number of apps available in leading app stores as of July 2014. http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores. 03/27/2015.

[So11] Soylu A., Wild F., Mödritscher F., Desmet P., Verlinde S., De Causmaecker P.: Mashups and Widget Orchestration. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystem (MEDES'11), San Francisco, 2011.

[So15] Smartphone OS Market Share, Q4 2014. http://www.idc.com/prodserv/smartphone-os-market-share.jsp. 03/27/2015.