End-User Creation of Social Apps by Utilizing Web-based Social Components and Visual App Composition

Juwel Rana, *Sarwar Morshed, Kåre Synnes
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
SE-97187, Luleå, Sweden
{juwel.rana, muhammad.morshed, kare.synnes}@ltu.se

ABSTRACT

This paper presents a social component framework for the SatinII App Development Environment. The environment provides a systematic way of designing, developing and deploying personalized apps and enables end-users to develop their own apps without requiring prior knowledge of programming. A wide range of social components based on the framework have been deployed in the SatinII Editor, including components that utilize aggregated social graphs to automatically create groups or recommending/filtering information. The resulting social apps are web-based and target primarily mobile clients such as smartphones. The paper also presents a classification of social components and provides an initial user-evaluation with a small group of users. Initial results indicate that social apps can be built and deployed by end-users within 17 minutes on average after 20 to 30 minutes of being introduced to the SatinII Editor.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services; H.4.3 [**Communications Applications**]: Social apps, Group-based apps;

General Terms

Mashup, Social apps;

Keywords

Component-based social app development, Tools for social app development, Social data, Mobile social app;

1. INTRODUCTION

Popular social media services such as Facebook[1], LinkedIn[2], Google+[3] and Twitter[4] are predominantly used today as web-based applications on personal computers or as apps on mobile devices such as smartphones [14]. The availability of these services drive the technical development where, for instance, smartphones without a significant number of apps are less attractive to users [10]. There is, thus, a strong incentive to make it easier to build and deploy social media services, while also offering possibilities for personalization and more effective social distribution mechanisms [26, 25].

Many social media services provide different tools for sharing presence information, sharing multimedia contents and performing collaboration activities [11, 8]. Facebook, Twitter, and Google+ all provide developer APIs for developing new apps. These APIs can also be used for integrating social features on services like Netflix[5], Spotify[6], and YouTube[7]. Netflix is a video streaming service that uses the Facebook platform APIs for user authentication as well as for sharing user experience when watching movies. Moreover, Facebook statistics show that by the end of 2011 there were more than 7 million social apps and websites that utilized the Facebook platform.

Today, developers are also able to design and develop mashup apps which typically contain more than one source of information to enhance the functionality or the presentation [27, 12, 16]. For example, police statistics and Google Maps can be integrated with a mashup app by combining the crime records for each city street.

However, users without programming knowledge are naturally often fall behind with respect to social app development. The main reason is that the social networking platform APIs are designed in such a way that code augmentation is necessary for adding social features to apps [1, 2, 7]. The focus of this paper is, therefore, on the following research question:

"How can end-users' social and communication data be captured from various data sources and then be utilized by the end-users to compose social apps?"

To answer this question, this research extends current mashup technologies for social apps in two ways. Firstly, it provides new tools and techniques for social app development through the SatinII App Development Environment, where the SatinII Editor allows end-users to visually compose social apps that are then compiled into web-based apps. This makes it easy to create personalized web-based social apps [6, 5] for communication and interaction that leverage the power of social computing APIs. Secondly, the editor hides the complexity of associating social networking APIs from the app developers' perspective, such that anyone

*Recently moved at Green University, Bangladesh
[1]http://www.facebook.com/
[2]http://www.linkedin.com/
[3]http://plus.google.com/
[4]http://www.twitter.com/
[5]http://www.netflix.com/
[6]http://www.spotify.com/
[7]http://www.youtube.com/

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

should be able to create a simple app. The social networking APIs are wrapped in components for this reason and developers create apps by putting components together visually. At present, 15 components for accessing and utilizing data related to social networks have been published in the SatinII App Developer Environment, including components that wrap Facebook and LinkedIn functionality [4].

This paper proposes a component-based approach to develop social apps, where so-called social components wrap fundamental functionalities of social media service APIs such as social data collection, social data visualization, text communication, and so on. For example, one component can retrieve birthdates from a user's social networking service and a messaging component sends text messages to a list of contacts. Thus, by composing these two components, a user is able to compose a social app for sending birthday wish messages (automatically) to the friends in his social network.

An important feature is that social components can provide new ways of managing contacts and initiating communication. For instance, using the social components proposed in this paper, users are able to generate mashups to visualize global contacts, forming groups by adapting contexts [26], and connecting social contacts for different interaction purposes [25].

The overarching aim is to minimize the gap between the traditional app developer and non-programmers, or more precisely, enabling end-users to develop social apps by using a component-based visual interface in a web browser while utilizing intuitive interaction techniques such as drag-and-drop. By doing this, developers are not required to know programming languages such as JavaScript, PHP, or HTML5. In other words, very complex functions can be made simple to use and thus enable end-users to create powerful personalized social apps. However, the SatinII Editor [3] itself is not the focus of this paper. Instead, we study the potential impact of end-users being capable of building apps based on social components.

This paper presents an initial study with 20 participants, with and without programming experience, that collects participant feedback from developing social apps using the SatinII Editor. We present three different social app development scenarios to the participants and measure different factors during the process, such as the average time for app development and the functionality of the composed apps. During the study, the average time taken for app development was 17 minutes, after 20 to 30 minutes of introduction, which indicates the potential impact of the social components and the SatinII Editor.

The rest of the paper is organized as follows: Section 2 provides related work, Section 3 describes the framework for social components, Section 4 presents an overview of the social components in the context of SATIN platform, Section 5 provides an evaluation and the results of the social component user study, and Section 6 provides a discussion and future work. Section 7 concludes the paper.

2. RELATED WORK

The literature shows there are several graphical tools [17, 27, 28] that are already available for making mashup apps. By using these tools, users are able to create different kinds of apps without specific programming skills. In most cases, users use the drag-and-drop interfaces of these tools to compose different visual objects or components to create mashup apps. However, none of these tools focus on social apps composition and there is no clear indication to improve these tools for social apps composition.

Liu et al. have proposed a mashup platform using service oriented architecture (SOA) [17]. The proposed architecture can be used by users to create their own service compositions. The authors provide a generic component model for the mashup architecture, which is generally similar to our component model. The model proposed in this paper focuses, however, on social components in particular.

Wong presents MARMITE as a mashup platform that offers a set of components and allows users to create their own mashup apps [27]. MARMITE allows users to extract information from one or more web pages, process or manage data flow and integrate data for different outputs in map-based services or text files. By comparison, the approach proposed in this paper utilizes data from the data providers' platform through proper authorization protocols that enables the users to build more personalized social apps.

iSocial describes important aspects of social networks and services composition on a mashup framework [18]. iSocial shows that social computing may influence sharing, competition and collaboration among the mashup framework users. However, it would be better if users were able to use their existing social networks information to influence sharing and collaboration. In our approach, we provide several components that retrieve information from the users' existing social networks and integrate that information in components for building personalized social apps.

Intel Mash Maker is a web extension to existing web browsers that allows users to expand the page with information from other websites [13]. A user can create a new mashup and add intelligence to the mashup with Mash Maker. After learning about the new mashup, Mash Maker suggests this new mashup to other developers.

Jung and Park propose an ontology-based mashup creation system that enables end-users to use different kinds of web-based data sources to construct a mashup [15]. This system utilizes a mashup rule language for combining content from multiple websites, which requires end-users to learn the rule language first in order to be able to compose mashup apps.

Marcio et al. propose a framework for building intelligent social apps by exploiting Facebook and FourSquare social network data [19]. The authors identify several challenges regarding using data from social networks, including privacy. They also describe social data as characterized by a large search space of user-generated data. Our proposed framework is able to deal with those issues since it is designed to adapt information of any kind from social networks and to support developers without requiring programming skills.

Berners-Lee et al. proposed a platform called Tabulator which links Resource Description Framework (RDF) data in order to create new apps [9]. Tabulator allows users to search an RDF graph in a tree structure, which enables Tabulator to create tables, Google maps, calendars, timelines, etc.

Morbidoni et al. present "Semantic Web Pipes" which is a powerful tool to create RDF-based mashups [21]. This tool aggregates and manipulates the content based on different RDF data. The semantic web pipe supports operation

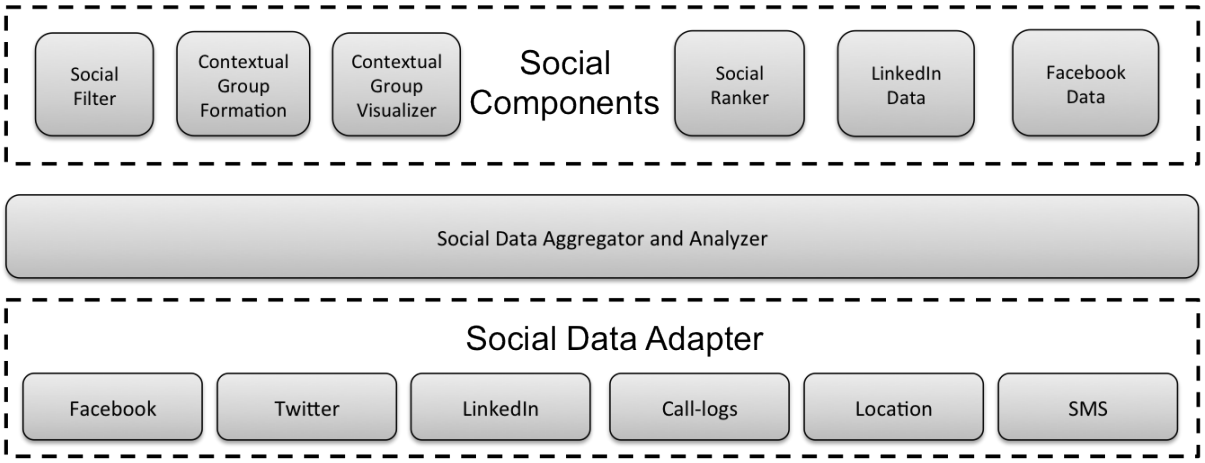


Figure 1: The Social Component Framework

ranging from straightforward aggregation to complex collaborative editing and the filtering of distributed graphs.

In the next section, we discuss the framework for social components, which illustrates a new way of utilizing heterogeneous social APIs from a single interface.

3. SOCIAL COMPONENT FRAMEWORK

Figure 1 shows a generalized and simplified framework for social components. The figure contains three different layers. On the bottom layer, social data sources are connected to fetch users' social networking data, for example Facebook data, Twitter data, LinkedIn data, and so on. Apart from the social networking services platform APIs, other APIs are used in this layer to fetch email logs, call logs, calendar events, and location logs. The middle layer provides a temporary storage of the users' social data, performs aggregation and analysis on those data to offer different functionalities such as ranking social contacts and so on. The methods that are used for processing and ranking user data are described in other papers [24, 23]. Actual social components perform from the top layer. For example, in Figure 1, the top layer shows components such as social ranker, contextual group visualizer, and so on, accessible by end-users. Here, all the components are shown in a generic manner. For example, the Linked Data Component and the Facebook Data Component fetch data from external sources and the Social Ranker Component and the Contextual Group Visualizer Component process those data by adding functionality. Figure 2 shows a classification model for social components. The SatinII Editor's interface can be considered as representative of the top layer in comparison to the social component framework.

Idea development for some the social components is inspired by the previous research work on Aggregated Social Graph (ASG) Service [23, 22]. The ASG service measures individuals' social strength by analysing communication history.

During component development, it is found that various components have different levels of complexity. For instance, in the case of social components, most of the components are based on external Web/REST APIs. Therefore, we find it important to identify different classes of components that need to be developed. We primarily classify two groups of components, namely *core components* and *supporting components*.

Core components are the main social components such as social filter, social data adapter and so on. Many of these classes of components have been created and deployed to the SatinII Editor. More information about different social components is available on the social distribution testbed website [4]. There are some components that help when triggering or labelling apps, which we called *supporting components*. In the component library of the SatinII Editor, TriggerButton and Lavel, are considered for generic purposes and classified as supporting components. We find it important to provide a classification model of components to help social app developers. The classification can be helpful for finding appropriate components for app development. Figure 2 illustrates the social component classification model.

The core components are sub-divided into eight different types of core components and are listed as follows:

- *Social Data Adapter Components*: This class of components adapts social data from social networking sites using the users' credentials. The data is stored in the Cloud for context reasoning.
- *Social Data Connector Components*: This class of components provides an interface to communicate between data sources and other components that utilize data.
- *Social Data Processing Components*: These components apply unified data representation [24] to enhance data mining within social data sources.
- *Social Data Reasoning Components*: These components implement different logic on the social data to a users desired apps.
- *Visualization Components*: Visualization components display different forms of social data. For example, if a processing and reasoning component makes a social group based on a user's social data, visualization components can show this group in grid view or graphical tree view.

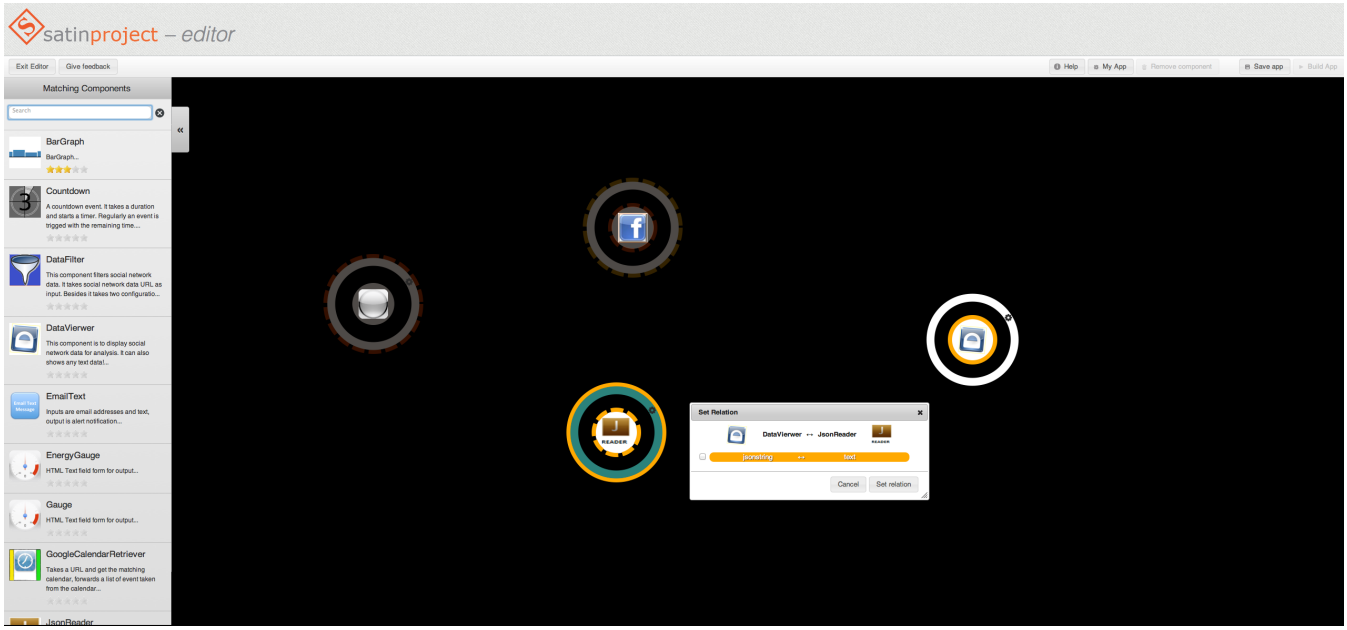


Figure 3: A sample social app composition in the SatinII Editor

- *Smart Object Components*: Smart object components provide interfaces for lightweight devices with messaging and web connectivity functionalities.

- *Messaging Components*: These components provide different options for sending messages such as email, posting to social networks, SMS, and so on.

- *Location-Based Components*: These components use location APIs and social network location-based services such as *Checkin*.

In the current model, supporting components are not divided into sub-categories. Since this paper focuses on social app development, the current model provides details only on social components.

The next section briefly covers the social components in the context of SATIN platform.

4. INTEGRATING SOCIAL COMPONENTS IN SATIN PLATFORM

As mentioned in Section 3, different kinds of core social components along with supporting components have been developed and simulated. Users social data are embedded as social data components and integrated with the SATIN platform where end-users without any programming knowledge could create their own social apps. Additionally, other social components embed intelligence that could exploit social data to create social apps. In our case, the SatinII Editor [3] is used as a testbed for simulating and evaluating the proposed app creation environment. The background of the SatinII Editor is beyond the scope of this paper, however details can be found online [3]. Social components are based on Web Technologies (i.e., HTML5, JavaScript, AXIS2 Web Services and so on), which enable users to run and test their apps regardless of the type of device. For example, any modern web browser (both on desktop and mobile devices) should be able to run SatinII-based mobile apps.

Data adapting (or collecting) components along with other supporting components aggregate users' different social network data. Data aggregating components understand the format of the stored social data and aggregate them as a single user's social data based on a particular context key or as a whole data stream. Later, this data resource can be analyzed and reused by data processing and analyzer components in order to create a user's personalized social apps. Data processing and analyzer components are embedded with specific intelligence to perform different kinds of data processing and analysis such as a user's social interactions, social media distribution, interests, location, etc. Data visualization components are used to show a user's aggregated social graph either in graphical form or in text form.

```
{   "Friendsname": "Test User",
"userid": 1323732759,
"username": "t.user",
"birthdate": "1982.02.20",
"email": "t.user@socialnetowrk.com",
"profileurl": "http://www.socialnetwork/t.user",
"movies": "ironman, avatar",
"interests": "programming, research, soccer",
"picture": "https://a.akama.net/3_t.jpg",
"contextkey": "stanford,stockholm,kth,ltu,"
}
```

Listing 1: JSON properties for profiling a friend's basic information

Different kinds of social data adapters (e.g., LinkedIn Data, Facebook Data, Gmail Logs, etc.) have been developed and tested in order to collect users' data from these data sources.

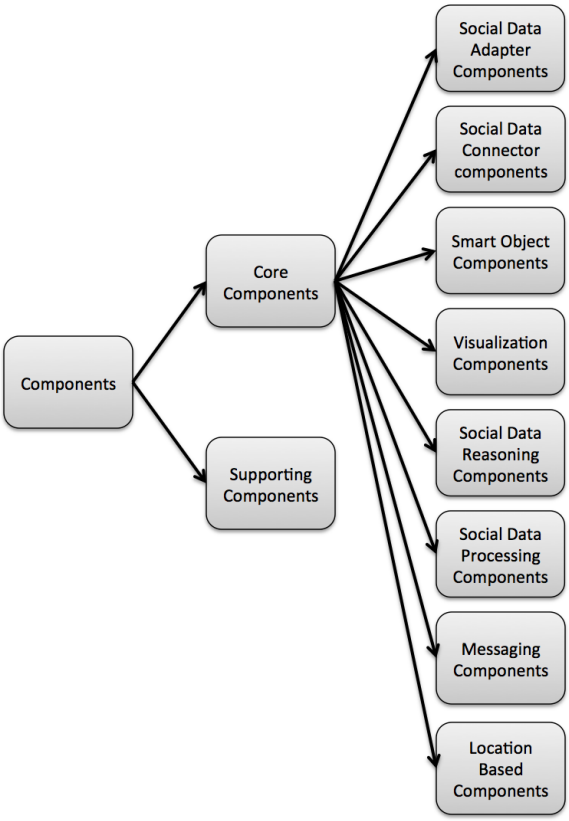


Figure 2: A classification of social components

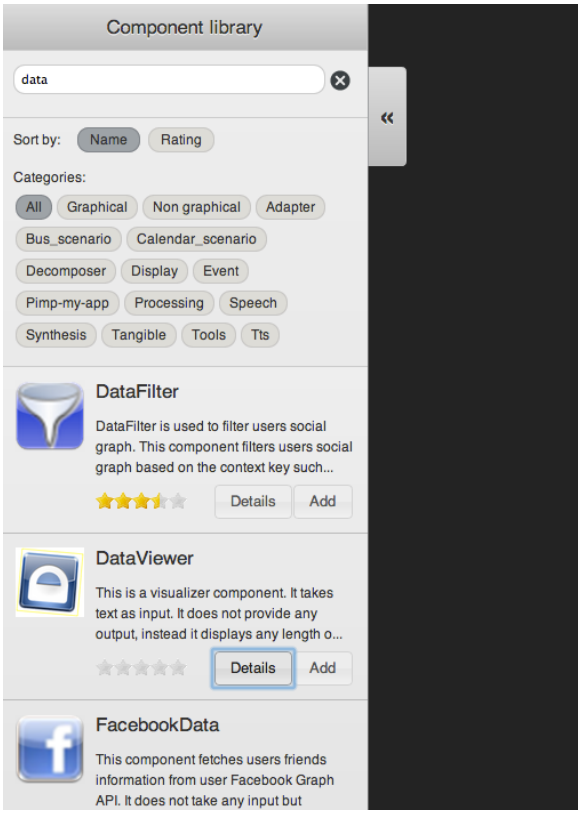


Figure 4: Components Library

In this implementation, all collected data from these data sources are stored in JSON properties. The same JSON data format is applied to all other data sources to solve user-specific data aggregation problems. Listing 1 shows the data format of profiling a friend's basic information through social components. The same data format is applied to Facebook data, Linkedin data, Google+ data, and so on.

In this regard, we also implement a JSON property reader as a supporting component to parse important information from users' social data. The app developer could easily filter the data properties according to the requirements of the desired app during the composition period. Three examples of social apps are shown in the evaluation section.

A large obstacle for component-based social app creation is social data aggregation and the transformation of these data for forming apps. Within deployed social components in the SATIN environment, there are components that aggregate data from different sources and provide a personalized data source. To implement this, a new indexing method based upon a user's access in multiple social data sources is considered. The JSON properties of the index file used to associate a user's multiple social identities and social data sources are shown in Listing 2. The social data visualizer component could be used to visualize social data. Another social component, called Social Data Filter, filters users and aggregates social data based on the filtering parameters. For instance, this Data Filter Component could be used to create groups with the user's social connections based on a user's interests. There are also other social components that are used to share a user's social resources with his/her connections.

```
{"index":[
  {"asguser":"1323732759","data":
{  "username":"t.user","socialIDs":
   {  "facebookId":"testuser12",
   "linkedinId":"user1",
   "googlePlusId":"testuser"
   },
   "socialDataPaths"
   {  "facebook":"../jfile/facebook/testuser12",
      "linkedin":"../jfile/linkedin/user1",
      "googleplus":"../jfile/googleplus/testuser"
   }
}
 }]}
```

Listing 2: JSON properties for associating a users multiple social identities

Figure 3 shows the composition of an app in the SatinII Editor. In the figure, one supporting component is used with three social components including Facebook Data, JSON Reader and Data Viewer. Users can drag and drop the components from the Component Library. The Component Library is shown in Figure 4. In this library, a user may find the components required for composing an app. For composition, the editor provides a canvas. A user needs to drag

Figure 5: Components composition area-Canvas

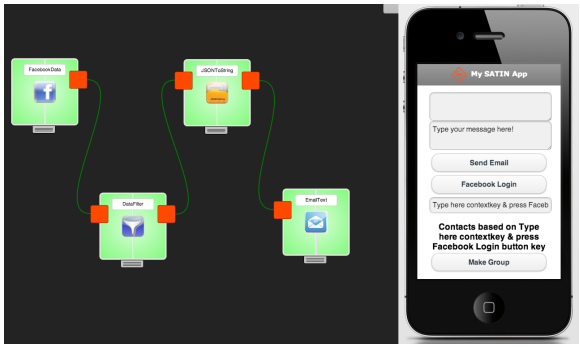


Figure 6: Composing components in the canvas and previewing the app

and drop the desired components on the canvas before composing an app. Figure 5 shows some of the components on the canvas. If components are able to be composed, then the user would be able to connect those components with an arrow. Before building the app, the user would be able to preview the app in the App Previewer. Figure 6 depicts an app previewer. The previewed app in Figure 6 is used to collect friends' contact information from Facebook, filter social contacts based on a particular context and send an email message to a group of filtered contacts in order to arrange a social event. The Facebook Data Component provides the functionality to fetch data from the Facebook platform by using the user's Facebook credentials. Then the mined data is transformed to JSON properties and stored in the SATIN server. The Data Filter Component is able to read the JSON properties and filter the data based on a user's given context information. After that, the JSONtoString Component fetches users' email addresses from the group of filtered contacts. The addresses are fetched by the EmailText Component. This component also contains a text box in which a user may add a text message and send it to the group. Thus, the app is used to invite specific friends on specific occasions to specific events. In a similar way, users can use other social data components such as Linkedin Data, Twitter Data or Google+ Data to fetch their social contact information from those social data sources.

The next section provides an experimental overview and the results of the user studies for social app development.

5. EVALUATION

We initiated a user study with a small group of users to get initial feedback on social app development from users, while not attempting to evaluate the whole SatinII App Development Environment. For that purpose, we selected users who have some knowledge about mobile apps, 10 with and 10 without programming experience. To run the study, we prepared three different scenarios which we gave to the users for social app development. Before the users started app composition, the available components for the social apps were also introduced to the users by providing written descriptions of the components as well as a demonstration on app composition using the SatinII Editor.

In the following subsection, descriptions of the scenarios are given.

5.1 Scenarios for the user study

The following three scenarios were provided during the user study to guide the end-users in developing corresponding apps with the SatinII Editor.

Scenario 1:
Bob is planning an outdoor party at an interesting, new place he has recently discovered. He wants to invite his Facebook friends to join the party from their current location. Thus, Bob shares an app on his Facebook timeline in order to invite his friends to join him, along with a map that directs them to the outdoor party from their current location.

Scenario 2:
Alice wants to know about the locations that have the greatest influence on her. So, she would like to make an app that automatically checks her current location and tracks it if she stays in that particular place for more than 30 minutes. At the end of the month, she would like to have stats about the places she stayed for more than 30 minutes in a ranked manner, meaning the most frequently visited places will appear first.

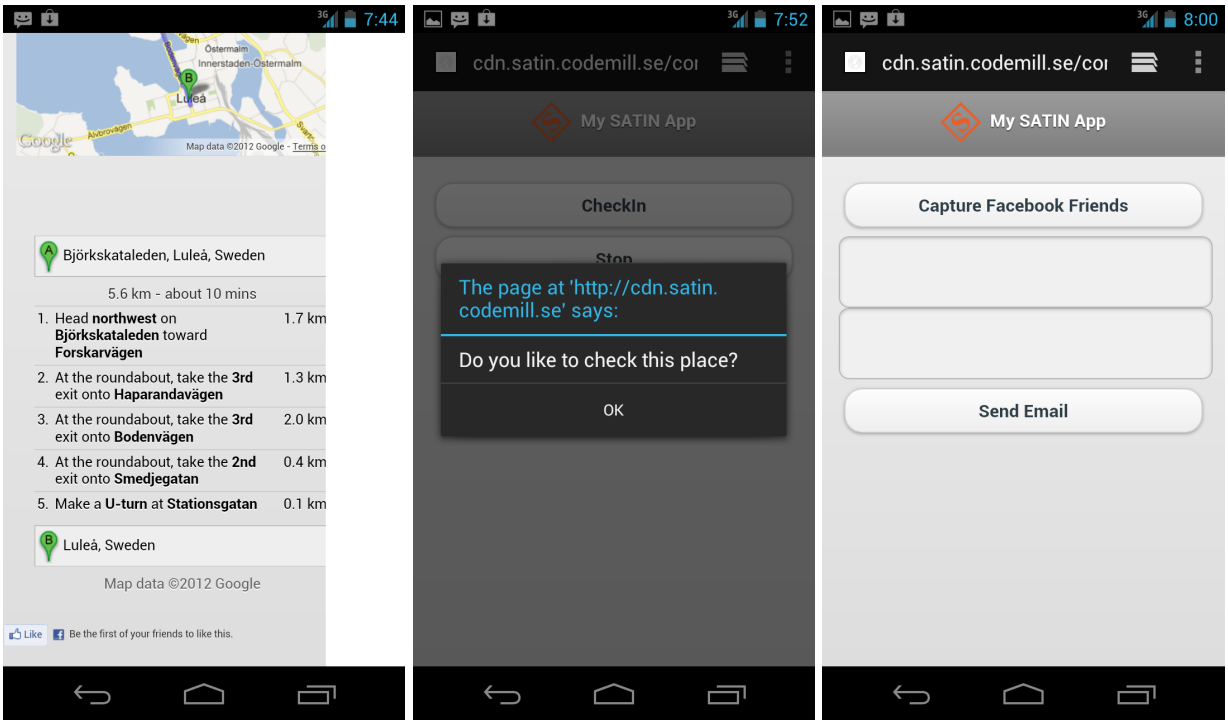
Scenario 3:
Charlie is going for a coffee break so he would like to send a message to his co-workers alerting them in case they want to take their coffee break at the same time.

5.2 Data Collection

As mentioned earlier, the study has been performed with two groups of 10 users. The first group of 10 users had at least some experience with computer programming and was selected from a population of staff and MSc students at Luleå University of Technology.The second group of 10 users had little or no experience with computer programming but had used apps on smartphones. They were selected from a population outside of the university.

The following parameters have been considered during the test for data collection:

- *App Composition Time*: The time duration that a user spent composing an app for a given scenario.
- *Number of Component*: The number of components that have been selected to perform composition.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 7: Sample social apps generated by the SatinII users

- *App Formation*: Whether the user is able to build an app and able to run the app after the composition.
- *App Functionality*: Whether the app composed by the user is functioning correctly with respect to the scenario in question.

Moreover, we collected the users' individual opinions based on the following parameters:

- *Social Acceptance*: Are the social component-based apps acceptable from a societal perspective?
- *Positive Affect*: Are the social component-based apps useful in making social interaction easier?
- *Quality of Experience*: Are the social apps user-friendly compared to available apps on smart mobile devices?
- *Control*: Do the social components provide sufficient control to compose different apps?
- *Ownership*: Does the ownership of the composed app remain with the app developer?

The opinion data was collected after the test users performed the above-mentioned task of app composition. The user provided ratings on a Likert scale from 1 to 5, where 1 is the most negative response and 5 is the most positive response [20].

Figure 7 shows a snapshot of three apps generated during the user tests. Figure 7(a) shows an app based on scenario 1. Figure 7(b) shows an app based on scenario 2 and Figure 7(c) is based on scenario 3. The functionalities available in the SatinII Editor do not fully comply with the described scenarios. For example, the app for scenario 1 shares an invitation through a Facebook-like operation, which could also be done with other options.

5.3 Evaluation Results

Figure 8 depicts the users' ratings of the social apps creation using the SatinII Editor. In general, we received positive responses from the majority of the users. Based on answers from the first user group who had some programming knowledge, the results show average social acceptance 4.2 (with standard deviation $\sigma = 0.7888$), average positive affect 4.7 ($\sigma = 0.4830$), average quality of experience 4.0 ($\sigma = 0.8164$), average control 3.6 ($\sigma = 1.0749$), and ownership 4.1 ($\sigma = 0.8755$).

For the second group of users who do not have programming knowledge, we received slightly different results. The results show average social acceptance 4.5 ($\sigma = 0.5270$), average positive affect 4.5 ($\sigma = 0.5270$), average quality of experience 3.6 ($\sigma = 0.5163$), average control 3.2 ($\sigma = 1.1352$), and ownership 2.9 ($\sigma = 1.1972$).

Figure 9 shows the time duration for the composition of the social apps. Although a significant amount of assistance was given to the users before or during app composition, the app composition time duration varies significantly from user to user. The average time is calculated at 16.6335 minutes ($\sigma = 7.6166$) required per user to compose apps based on the given scenarios. However, in comparison between the two groups of users, it is found that the group with program-

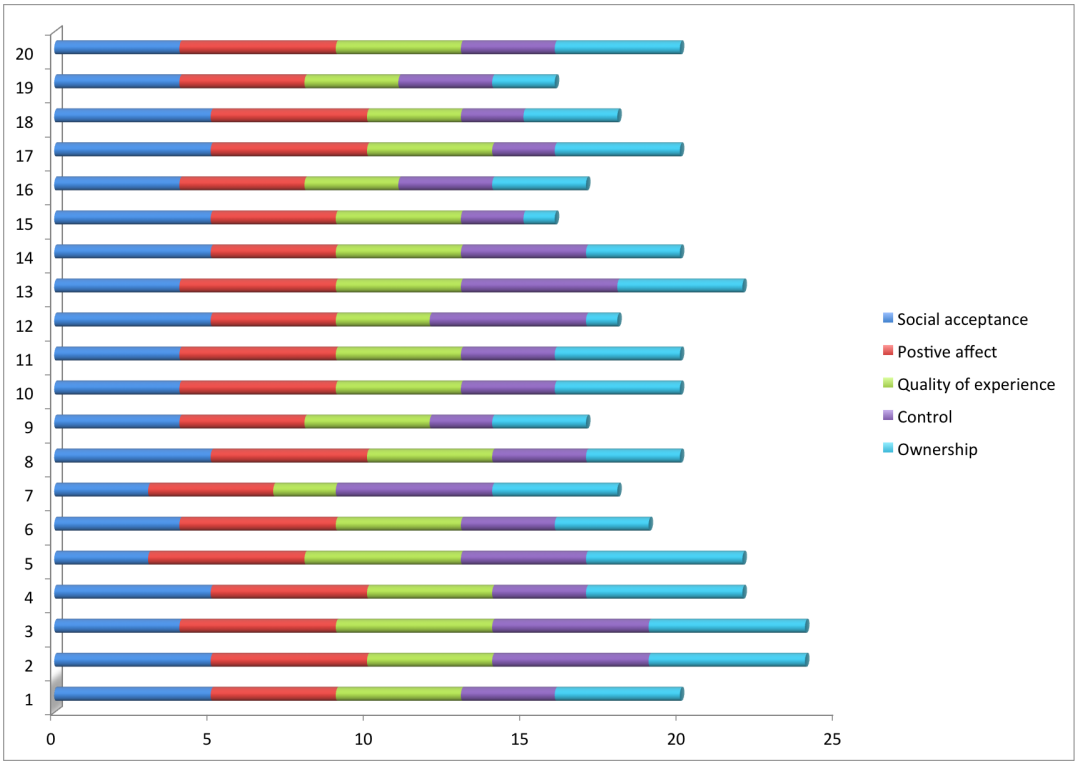


Figure 8: Individual user ratings of the social components framework, presented using a 5-point Likert scale based on social acceptance, positive affect, quality of experience, control and ownership. The vertical axis indicates the users index and the horizontal axis indicates ratings of the users.

ming knowledge takes 10.967 minutes ($\sigma = 4.8688$) while the group without programming knowledge group takes 22.3 minutes ($\sigma = 5.2357$).

A common problem for most of the users during app composition is that they were not comfortable enough with the composition environment and asked for support in identifying appropriate components to implement desired functionalities in their apps. This feedback from the users was not completely unexpected and will be used to improve the composition environment. However, this is not directly connected to the social component framework, but for the SatinII Editor in general. The next version of the editor will be enhanced with several functionalities to help users identify components (for instance filtering and recommending components based on previous use). The positive impression from this user study is that after being able to create an app, the users were relaxed and appreciated the environment as well as their personalized social apps. The next section discusses the research question and provides ideas for future work.

6. DISCUSSION AND FUTURE WORK

This paper proposes a framework for social components that is tailored to the SatinII App Development Environment.

The component development model mainly adheres to the social aspects of app development. It identifies different core and supporting components used to compose social apps. The paper also contains a classification of components, to give an overview about the kinds of components that could add value to the mobile app development environment. We argue that there are a wide variety of components that need to be developed and classified for diverse app development, and that is one of the targets for the future work.

From the user study, we found that there are difficulties in understanding the composition scheme of the SatinII Editor. However, with initial support from an instructor, the users could understand the app composition methodology within 20 to 30 minutes. Our evaluation provided useful feedback from the users despite the fact that the scale of the user study was limited to 20 participants.

"How can end-users' social and communication data be captured from various data sources and then be utilized by the end-users to compose social apps"

The research question addressed in this paper considers social data as one of the important areas of component development. Facebook and LinkedIn data components attracted users, since they were interested in developing social apps based on those two platforms. In this paper, it is shown that the social component framework provides a standard way of developing social components to capture data from social media sources. It also shows the different types of social data collector components developed for the SatinII platform. The framework could therefore be used as a model for other social component developers, who could

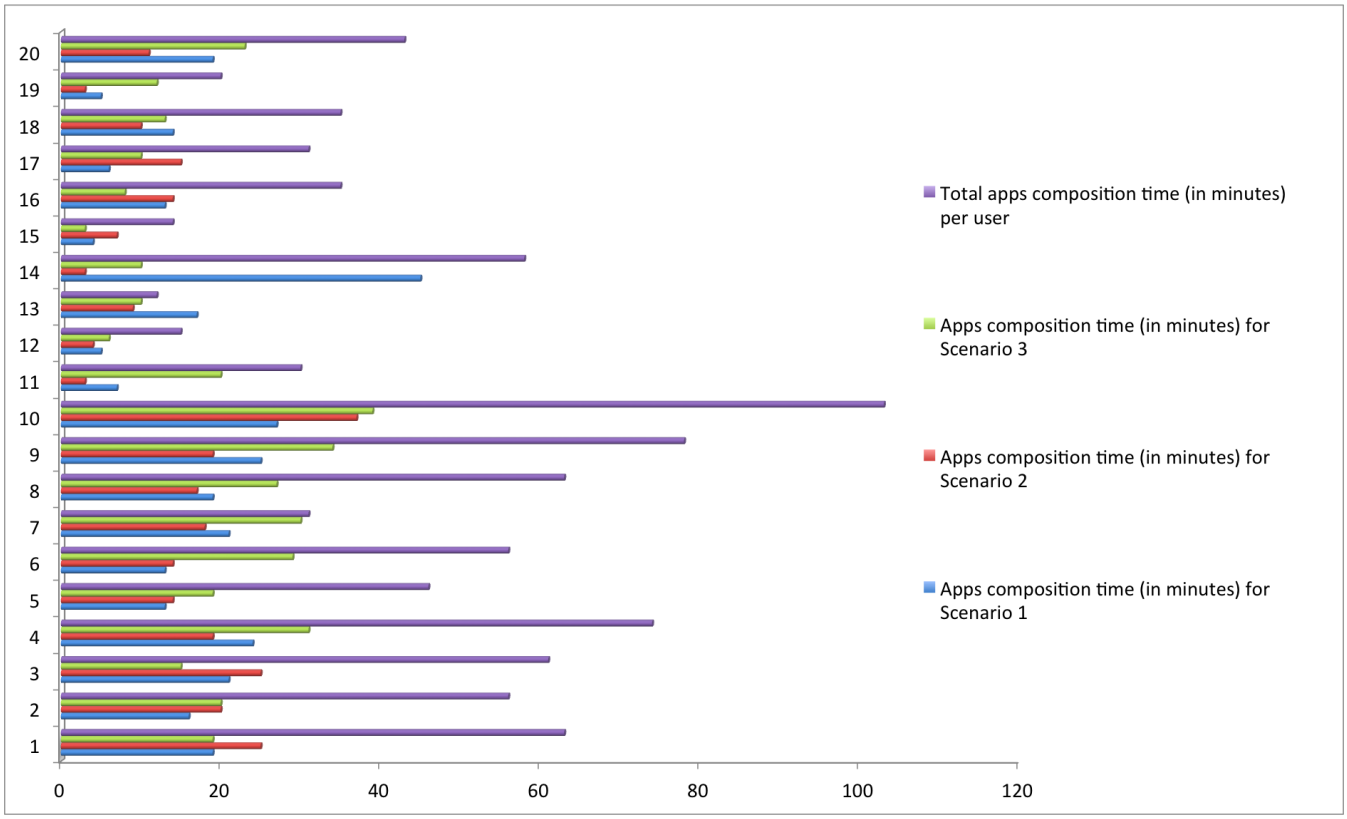


Figure 9: Social apps composition time on a user basis considering three different composition scenarios. Vertical axis indicates user's index and horizontal axis indicates time in minutes.

adapt our JSON properties to enable their components to be compiled using SatinII-based social data components. New component developers can thus benefit from designing and implementing their social components for the SatinII App Development Environment. The potential aspect of future work is to cover different domains of app developments such as games, mobile OS-based native apps and so on.

7. CONCLUSIONS

The main contribution is that end-users, without specific programming skills, can easily create social apps by utilizing social components and a visual editor. The study shows that the implementation of an application from any of the three simple scenarios took on average 17 minutes after a 20 to 30 minute introduction. This clearly indicates the potential impact of the presented environment, as the study shows that end-users without practical knowledge in programming could easily create personalized social apps for media and other distribution purposes. The results also show that the two test user groups are distinct from each other, where non-programmers need double the amount of time to complete compositions.

A final conclusion is that while smart devices and social apps are becoming a part of everyday life, the potential for utilizing personalized social apps for media distribution, group formation, lightweight collaboration and so on is great. This will potentially open up for new and more efficient ways of social interaction.

8. ACKNOWLEDGEMENT

The work was supported by the Centre for Distance Spanning Technology (CDT) at Luleå University of Technology and by the SatinII research project, funded by the European Regional Funds (mål-2), the Swedish Agency for Economic and Regional Growth, the County Administrative Board of Norrbotten, Norrbotten County Council, and the City of Luleå. The authors thank the SATIN core-tech group and the SATIN design group.

9. REFERENCES

[1] Facebook Developer Tools. `https://developers.facebook.com/docs/`, 2012.

[2] LinkedIn Developer Tools. `https://developer.linkedin.com/`, 2012.

[3] SATIN Editor. `http://www.satinproject.eu/`, 2012.

[4] Social Distribution Testbed. `https://sites.google.com/site/socialdistributiontestbed/`, 2012.

[5] Social Software. `http://en.wikipedia.org/wiki/Social_software`, 2012.

[6] Social Web. `http://en.wikipedia.org/wiki/Social_web`, 2012.

[7] Twitter Developer Tools. `https://dev.twitter.com/`, 2012.

[8] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer. Falling asleep with angry birds, facebook

and kindle: a large scale study on mobile application usage. In *Proc. of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 47–56, 2011.

[9] P. D. Bra, A. Aerts, B. Berden, B. de Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash. Aha! the adaptive hypermedia architecture. In *Proc. of the fourteenth ACM conference on Hypertext and Hypermedia*, pages 81–84. ACM Press, 2003.

[10] E. Castledine, M. Eftos, and M. Wheeler. *Build Mobile Websites and Apps for Smart Devices*. Sitepoint, 1st edition, 2011.

[11] Y. Cui and M. Honkala. The consumption of integrated social networking services on mobile devices. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, MUM '11, pages 53–62. ACM, 2011.

[12] F. Daniel, M. Imran, F. Kling, S. Soi, F. Casati, and M. Marchese. Developing domain-specific mashup tools for end users. In *WWW (Companion Volume)*, pages 491–492, 2012.

[13] R. Ennals, E. A. Brewer, M. N. Garofalakis, M. Shadle, and P. Gandhi. Intel mash maker: join the web. *SIGMOD Record*, 36(4):27–33, 2007.

[14] C. Faloutsos and U. Kang. Managing and mining large graphs: patterns and algorithms. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 585–588. ACM, 2012.

[15] H. Jung and S. Park. Mashup creation using a mashup rule language. *J. Inf. Sci. Eng.*, 27(2):761–775, 2011.

[16] W. Kongdenfha, B. Benatallah, J. Vayssière, R. Saint-Paul, and F. Casati. Rapid development of spreadsheet-based web mashups. In *WWW*, pages 851–860, 2009.

[17] X. Liu, Y. Hui, W. Sun, and H. Liang. Towards service composition based on mashup. In *IEEE Congress on Services*, pages 332 –339, july 2007.

[18] X. Liu, N. Jiang, Q. Zhao, and G. Huang. isocialmash: Convergence of social networks and services composition on a mashup framework. In *Proceedings of the 2011 IEEE International Conference on Service-Oriented Computing and Applications*, SOCA '11, pages 1–6, Washington, DC, USA, 2011. IEEE Computer Society.

[19] M. E. F. Maia, J. B. F. Filho, C. A. B. de Q. Filho, R. N. S. Castro, R. M. C. Andrade, and F. Toorn. Framework for building intelligent mobile social applications. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 525–530, New York, NY, USA, 2012. ACM.

[20] A. Möller, S. Diewald, L. Roalter, and M. Kranz. MobiMed: Comparing Object Identification Techniques on Smartphones. In *NordiCHI 2012*, pages 31–40, Copenhagen, Denmark, Oct. 2012. ACM.

[21] C. Morbidoni, D. Le Phuoc, A. Polleres, M. Samwald, and G. Tummarello. Previewing semantic web pipes. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 843–848, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] J. Rana. *Improving group communication by harnessing information from social networks and communication services*. PhD thesis, avh. Luleå: Luleå tekniska univ., 2011.

[23] J. Rana, J. Kristiansson, and K. Synnes. Enriching and simplifying communication by social prioritization. In *Advances in Social Networks Analysis and Mining (ASONAM), Odense, Denmark*, pages 336–340. IEEE, 2010.

[24] J. Rana, J. Kristiansson, and K. Synnes. Modeling unified interaction for communication service integration. In *The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 373–378, 2010.

[25] J. Rana, J. Kristiansson, and K. Synnes. Dynamic media distribution in ad-hoc social networks. In *SCA2012, Xiangtan, China*, pages 546–553. IEEE, 2012.

[26] J. Rana, J. Kristiansson, and K. Synnes. Supporting ubiquitous interaction in dynamic shared spaces through automatic group formation based on social context. In *ASE International Conference on Social Informatics, Washington D.C., USA*. IEEE, 2012.

[27] J. Wong. Marmite: Towards end-user programming for the web. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 270 –271, 2007.

[28] Yahoo. Pipes. `http://pipes.yahoo.com.pipes`, 2010.