The Activity Platform

Helen J. Wang, Alex Moshchuk, Michael Gamon, Mona Haraty, Shamsi Iqbal,
Eli Brown, Ashish Kapoor, Chris Meek, Eric Chen, Yuan Tian, Jaime Teevan
Microsoft Research

***Abstract*—In this paper, we advocate "activity" to be a central abstraction between people and computing instead of applications. We outline the vision of the activity platform as the next-generation social platform.**

I. Introduction

"Applications: what a terrible term. What a terrible concept. ... We don't do word processing ... People do activities, and the software ought to support this." – The Invisible Computer, by Donald Norman 1998

Social, mobile, and cloud computing has brought unprecedented amount of information as well as much digital clutter to people today. Many people feel overwhelmed and burdened by organizing and finding information. Worse, "applications", the central abstraction between people and computing today, siloes a person's information, multiplying his/her efforts in organization and search. For example, information of one's project is often scattered across different applications, such as email, IMs, software repositories, web browsers, local folders, and remote folders. When searching information about the project, a user often has to go to many of these different silos to find it.

This problem calls for a fundamentally new abstraction between people and computing. In this paper, we advocate "activities" to be this abstraction. An *activity* is an ongoing effort in a person's life towards a goal. An activity can be a project for research, development, testing, product planning or event organization, a trip, or a hobby like book club. People already think in terms of activities. For example, when one wakes up in the morning or walks into her office, she often thinks about what needs to be done for each activity of hers that day. So, the activity abstraction is simply a representation of what is already in people's mind. We envision the activity abstraction to live on top of existing applications, leveraging and invoking their functionality in an activity-oriented way. For example, news feed of an activity can come from different applications including email, Facebook/Twitter feeds, or updates on documents; a todo list of an activity may contain bugs to be fixed from software project management tools or other sources; bookmarks may come from browser's bookmarks or command shortcuts on a local computer. To this end, activity needs to *interoperate* with existing applications.

Furthermore, applications can offer more superior user experience when they are *activity-aware*, as the activity context enables more personalized services. For example, today's shopping services like Amazon have used only one's shopping history to make recommendations; when with the activity context, they can string together the user's activity-specific shopping history and make more targeted recommendations for the user's activity. Operating system design can similarly benefit from the activity context. Instead of offering an application-centric operating system shell (as in all existing operating systems), an OS can offer an activity-based shell, as shown in Figure 1, letting users to directly view and interact with their activities and to dive into an activity context which includes the display state (which applications are openned with which files) and application state of each application in that activity. We hope it is evident

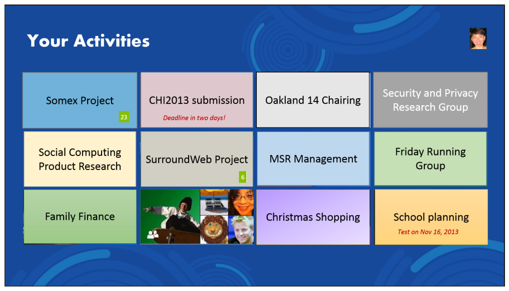


Fig. 1. **The Activity Shell**

now that to support the activity abstraction, we need an activity *platform*, which synthesizes activity-specific artifacts from each application and in turn supplies the activity context to applications.

Because many activities involve multiple people, we must build this social context into the activity abstraction and platform to allow collaborative curation for an activity. For example, an activity partner bookmarks a site for the activity, the other partners can see the bookmark as well. In another example, when an activity partner uses a search engine to search related literature for the activity (e.g., a research project). An activity-aware search engine can annotate each search result with whether my activity partner has clicked on the link or even whether they have written notes for it. Lastly, two parents are doing online Christmas shopping. When one bought a present for their children, an activity-aware shopping site can help the other partner see the presents bought so far to avoid buying duplicate presents.

The activity platform can be viewed as the next-generation facebook-like social platform. Today, Facebook offers applications the social context of a user, namely the user's friends and what the user and friends have shared on Facebook. In turn, applications contribute back a facebook user's actions in the applications (because it could potentially give the applications viral growth). Analogously, an activity platform offers applications the activity context, the users activities, their current activity, their activity partners, past actions in activities from a user or his/her partners. In turn, applications contribute back a user's actions for an activity. Facebook's Open Graph is the representation of social context, which is an entity graph that consists of entities (users or any objects) and their relationship as edges between them. An activity platform augments the Open Graph with "activity" entities to which existing entities belong, as shown in Figure 2.

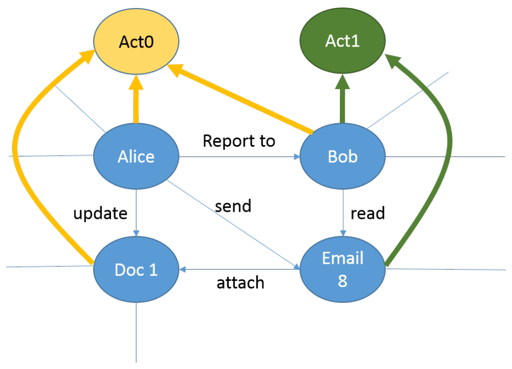


Fig. 2. **Activity-augmented entity graph**

II. Universal Tag and Collaborative Tagging

We introduce *universal tag* as a key mechanism to implement activity. Today, tags or labels have been well used in applications silos. For example, Gmail has used labels to organize emails, Google Doc uses folders, and web browsers use favorite folders to organize bookmarks. A universal tag unifies all these silo-ed labeling into a single one.

A universal tag is *access control-agnostic*, as illustrated in Figure 3. The data a user has access to falls into three categories: (1) privately accessible to the user, (2) group-accessible to the user because (s)he is part of a group, or (3) publicly accessible data (such as public web content). When an activity tag is applied to a data item, the tag does not change the access control of an item.

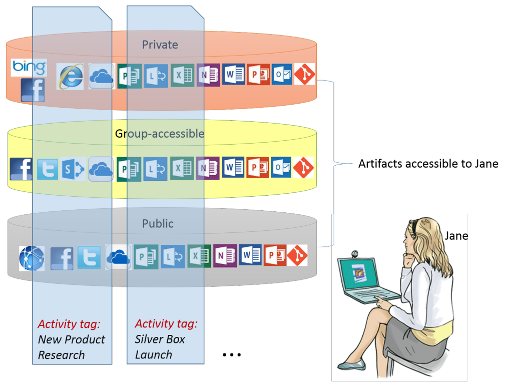

Private
Group-accessible
Public
Artifacts accessible to Jane
Jane
Activity tag: New Product Research
Activity tag: Silver Box Launch
...

Fig. 3. **Universal Tag**

This may seem obvious, but its subtlety has significant implications.

Access control-agnostic tagging is what makes the activity abstraction fundamentally different from the "group" abstraction as seen in Facebook groups or Yammer groups or Slack's channel. Making an item part of a group makes that item accessible to all the group members. In fact, the group accessible artifacts in the figure contain items in groups or channels.

There are still two key questions to answer: who can give an activity tag and who can see the activity tag.

- Who can give activity tag: Given that the primary goal of the activity abstraction is to help users organize their digital artifacts by their activities, it is undesirable for anyone to be able to tag an item to be associated with an activity. This can clutter a user's view of their own activites. So, we only allow activity partners to give the activity tags.
- Who can access the activity tag: There are three possibilities: (1) It can be private to the user who tags the item; (2) the tag can be visible to activity partners; (3) the tag can inherit the access control of the item. Careful readers would have dismissed (2), as the item may not be accessible to all activity partners since the activity tag is access control-agnostic. Between (1) and (3), (3) gives the great benefit of collaborative tagging, namely, when one activity partner tags a shared item, other partners who have access to the item can also see the tags so that they do not need to tag the item themselves. Collaborative tagging enables collaborative curation and organization of the activity for all partners.

We now address the differences between activity tags and hash tags that were popularized by Twitter. Hash tags can be created by anyone and used by anyone to tag (Twitter feeds for example). There is no-renaming of a hash tag, but one can always create a new one. Inevitably, there can be much ambiguity in hash tags. For example, #apple could be used to tag items related to fruit apple or related to the Apple products. In contrast, activity tags can only be created by activity partners (at the activity creation time). The activity tags are much less ambiguous as they are created and used only by activity partners. Activity tags are also rename-able, in the sense that activity partners can change the activity name at any time. This can be useful for example when a project name evolves. Fundamentally, activity tags and hash tags have different goals. Activity tags are intended for personalized, collaborative activity curation while hash tags are used for public curation of a hash tag. So these two mechanisms are orthogonal and can and should co-exist.

Like any tags used before, activity tags represent user labels that can be leveraged by the system to make recommendation on other items that may belong to the activity, helping users populate activities.

III. Challenges

In this section, we illustrate some challenges in designing the activity abstraction.

Visual representation of activities: Now that activities synthesize artifacts from various applications, the challenge lies in presenting these different types of artifacts to the user in an activity. With the ever increasing number of applications and unpredictable types of artifacts, we need a way to map any artifacts into a fixed number of categories to present to the user.

Minimize user burden: To make activity usable, we must minimize user burden in creating, bootstrapping and maintaining activities. We must make activity creation easy. Once the activity is created, we want to minimize the amount of tagging the user has to do to benefit from the system. For example, an activity can be created from an email or meeting calendar event where people involved become activity members and the content of the item can be used by the machine learning component of the system to immediately make useful recommendations

Protecting user privacy in the activity platform: Protecting user privacy is paramount in the activity platform, since activity data is among the most private data of a user. An application must obtain user permissions to access a user's activity data. Designing permissions systems for the activity platform (as well as social platform like Facebook) poses additional challenges to that of modern client OSes (such as the permission systems for Android and iOS): Permission systems for modern client OSes is to let the user of a device to grant permissions to applications to access sensitive resources on (and local to) the device, such as camera, location or address book. In the activity platform or social platform, because the user context is accumulated from different applications' contributions as well as from other users' contributions (such as a status update from the user's friend concerning the user), permission system designers need to be concerned about (1) whether an application's contribution should be accessible to another application, e.g., should Lowes know that the user has purchased a John Deere lawn mower from HomeDepot? (2) whether a user's contribution can be accessed by another user's application, e.g., can a user's TripIt app access the user's friends' liked cities?

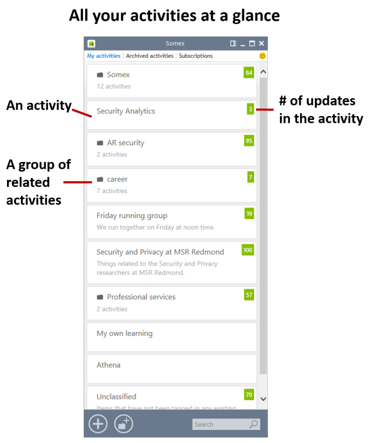
IV. Initial Design and Early Experience

We have built our initial design into an activity browser called Somex, which currently interoperates with Outlook, SharePoint, and Lync instant messenger. We also build an activity platform that exposes Facebook-like Open Graph APIs to applications with a focus on the permission system design.

We made an early deployment of the Somex activity browser from September 5-18, 2013 with 15 user using the tool. While we cannot draw any conclusions from such a small study, some of the results shed light on certain design decisions.

We are only scratching the surface in understanding and designing the activity abstraction and the activity platform. We describe future work respectively.

A. *Visual presentation of activities*


All your activities at a glance
An activity
A group of related activities
# of updates in the activity
Somex
My activities
Archived activities
Subscriptions
Search

Somex 84
12 activities
Security Analytics 3
AR security 95
2 activities
career 7
7 activities
Friday running group 19
We run together on Friday at noon time.
Security and Privacy at MSR Redmond 100
Things related to the Security and Privacy researchers at MSR Redmond.
Professional services 57
2 activities
My own learning
Athena
Unclassified 70

Fig. 4. **The Somex Activity Browser: Activity Index**

The Somex activity browser presents a high-level view of all the activities of a user, as shown in Figure 4. To more compactly present the data from different applications within an activity and to scale with increasing number of and different type of applications, we classify all entities out there into five basic types, as shown in Figure 5: *news feed* (e.g., e-mails, Facebook news feed, updates on documents, code or bug fixes), *todo list* (e.g., todo item from Outlook, bugs to be fixed from Github), *Documents and links* (e.g., links or shortcuts to remote files or local files, email attached documents), *Calendar Events*, and *People* who are involved in the activity.

Future work: It is an open question whether these five types can be sufficient for all the application types out there; it is future work to interoperate with many applications to understand this fully.

B. Minimizing user burden

We minimize user burden in creating and maintaining activities in the following ways.

Collaborative activity creation: Activity creation in Somex is collaborative. Once an activity parnter creates a project, other activity partners see that activity in Somex as well; this can significantly reduce project creation effort for activities with multiple partners. From our study, the 15 users would have created 89 activities, but with collaborative activity creation, they only need to create 43 projects, saving that effort for 46 projects.

Collaborative activity tagging: As described in Section II, activity tags on a shared item are visible to activity partners who also have access to the item. Therefore, when one partner populates an activity by associating an item with the activity, the shared item is also populated at other partners' activities if they can access the item. Our study shows that for 12 of 15 users, collaborative activity tagging populated more than half of their artifacts in their activities.

Machine learning suggestions: Somex uses a basic Linear Support Vector Machine (SVM) algorithm to suggest items that belong to an activity, using the already-tagged items as labels. For privacy, we perform only user-private machine learning. In other words, we do not perform machine learning globally across users. In this scenario, collaborative tagging significantly increases the number of labels for our machine learning algorithm and consequently improves the quality of the suggestions.

Future work: New approaches are needed in having the system to give high quality recommendations even with very few labels. It is also desirable that the system suggests new activities to users.

C. Permission system design for the activity platforms

We have analyzed the existing permission system of Facebook. We find that it falls short in the following aspects: (1) Many permissions are not least-privilege. Once the permission is granted to an application, the application has permanent access to the data including future data the nature of which the user cannot predict. For example, an application can get all the user's future check-ins once obtaining the User/CheckIn permission; an application can get all the user-liked items once obtaining the User/Like permission. From our 300 Mechanical Turk user study, we found that around 50% of users do not expect such future, permanent access. (2) The permission granting process incurs significant time overhead to developers. 36 out of 39 permissions require manual review and approval which takes from 3 to 14 days. (3) Certain sensitive permissions are denied, sacrificing functionality for privacy. "Read-Stream" and friends' sharing are deprecated permissions for better privacy. Nevertheless, some desirable functionality would be sacrificed. For example, when shopping for a wedding present at Macy Online store, the user would benefit from knowing what presents have been bought so far not just from Macy's, but also other online stores.

Learning from these lessons, we identify the following goals for a permission system of an activity platform: (1) Minimize the number of permissions to reduce developer and user burden. (2) Minimize the number of user prompts to reduce user burden. (3) Achieve least privilege to protect user privacy. (4) Enable cross-user, cross-application sharing without compromising the previous three goals.

To meet these goals, we employ the following three mechanisms in our activity platform: (1) Opaque handle. An activity platform can give an opaque handle of a piece of context to an application. The handle is the same across different sessions of the application. This way, although the application cannot see the context, it can collate the user's in-application behaviors across sessions to gain application-specific context for the user. For example, instead of providing a user's activity to an application, the activity platform could give an opaque handle of the user name to the application. The application can track the user behaviors across sessions without knowing who the user is. (2) Opaque display. Instead of passing a piece of privacy-sensitive context to an application, we pass an opaque display (such as a cross-domain iframe) containing the context to the application. This way, the user can see the activity data (possibly contributed from another application), but the application cannot access the data. Consider an activity of wedding registry in which wedding guests collaborate to buy wedding presents and want to avoid duplicate presents across all online shopping venues. At an online shopping venue, an opaque display can show the list of presents bought by other guests at other stores. This way, the online shop does not need to get permission to access that data which may contain privacy-sensitive data of a user's shopping experience at its competitors. (3) User-driven access control [4]. Let the user's natural interaction with an application grant the proper permissions. For example, a user shares a piece of context from an application by explicitly clicking on the share button provided by Facebook, building the permission into the design of the functionality of sharing in the application.

By using these mechanisms as much as possible, our

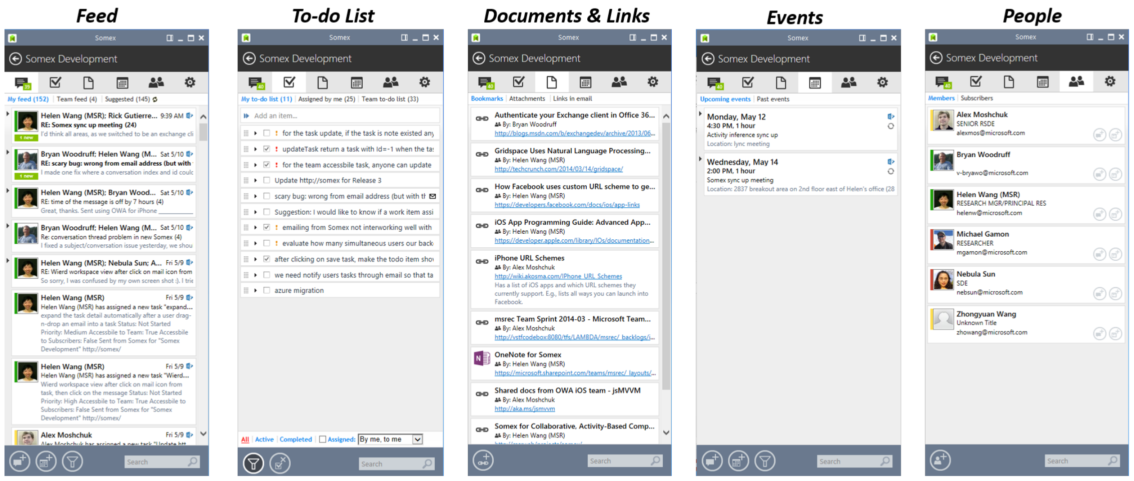


Fig. 5. **Five essential entity types in Somex**

permissions system needs just 1 permission compared with Facebook's 38 permissions today.

Future work: It is future work to study real-world applications that use Facebook's social platform and measure the impact of adopting the three mechanisms on them.

V. Related work

Bartram et al. (2003) built the ABC (activity-based computing) system for grouping related documents into projects for convenient project management and switching [1]. Likewise, Smith et al. built the groupbar, which grouped related windows into groups in the familiar taskbar [5], and Robertson et al. designed Scalable Fabric similarly, though leveraging the periphery and scaling for very large displays [3]. IBM's Activity Explorer [2] also uses activity to organize collaboration, but artifacts of an activity are always shared among activity members; furthermore it does not interoperate with existing applications, but has its own (siloed) artifacts.

VI. Concluding Remarks

In this paper, we have outlined the vision of making "activity" as a central abstraction between people and computing, which allows people to visualize, interact, and organize all their data by activity across application silos. Activities also represent critical context for applications to offer superior, more personalized user experiences. Fundamentally, an activity platform is needed to supply the activity context and to synthesize in-application activity data. The new, foundational concept we bring forth is *universal tagging*, which crosses application boundaries and is access control-agnostic.

We gave a brief description of our first design to address the three key challenges in designing the activity abstraction: We identify five basic types of entities to allow a compact and scalable presentation of activity data synthesized from unpredictable number of applications. We employ collaborative activity creation, collaborative tagging, and machine learning suggestions to ease the user burden. Protecting user privacy in the activity platform requires rethinking permission system design, in which we advocate the techniques of opaque handle, opaque display, and user-driven access control to minimize the number of permissions and the number of user interactions needed.

Many open questions remain across different research areas, including exploring new and tailoring existing machine learning algorithms to minimize user efforts, understanding the impact of activity entity in an entity graph for information retrieval, exploring user interaction model for accessing and finding related artifacts and activities, and activity-based operating system resource management and security policies.

References

[1] Jakob Bardram, Jonathan Bunde-Pedersen, and Mads Soegaard. Support for activity-based computing in a personal computing operating system. In *Proceedings of CHI*, 2006.

[2] W. Geyer, M. J. Muller, M. T. Moore, E. Wilcox, L.-T. Cheng, B. Brownholtz, C. Hill, and D. R. Millen. Activity explorer: Activity-centric collaboration from research to product. In *IBM SYSTEMS JOURNAL, VOL 45, NO 4*, 2006.

[3] G. Robertson, E. Horvitz, M. Czerwinski, P. Baudisch, D.R. Hutchings, B. Meyers, D. Robbins, and G. Smith. Scalable fabric: flexible task management. In *Proceedings of AVI*, 2004.

[4] Franziska Roesner, Tadayoshi Kohno, Alexander Moshchuk, Bryan Parno, Helen J. Wang, and Crispin Cowan. User-driven access control: Rethinking permission granting in modern operating systems. In *IEEE Symposium of Security and Privacy (The Oakland Conference)*, 2012.

[5] Greg Smith, Patrick Baudisch, George Robertson, Mary Czerwinski, Brian Meyers, Dan Robbins, Eric Horvitz, and Donna Andrews. Groupbar: The taskbar evolved. In *Proceedings of OZCHI 2003*, pages 34–43, 2003.