co-Laevo

Supporting Cooperating Teams by Working 'within' Shared Activity Time Lines

Steven Jeuris
Paolo Tell
Jakob E. Bardram

IT University Technical Report Series **TR-2016-193**

ISSN 1600–6100 **June 2016**

ISSN 1600–6100

ISBN 978-87-7949-358-2

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: **+45 72 18 50 00**
Telefax: **+45 72 18 50 01**
Web **`www.itu.dk`**

June 24, 2016

Abstract

In this paper, we describe the interaction design and implementation of co-Laevo: an activity-centric desktop computing system supporting task coordination within cooperating teams. Coordination is supported by having all team members orchestrate their dependent tasks on a shared activity time line. Each task (or activity) is associated with a personal dedicated workspace within which related resources, like files and windows, are embedded. As users access activity workspaces, the past, current, and planned state of the activity can be updated to reflect ongoing work. In contrast to stand-alone scheduling tools, like a team calendar, the actual work is not disconnected from the plan. In essence, users work 'within' a shared schedule, suspending and resuming activity workspaces in order to access the contained resources required for their work. Users are thereby constantly confronted with their own and collaborators' activities as part of task switching during everyday work. We introduce this concept, and several entailing design implications, as *cooperative activity life cycle management*. We anticipate the design of such a system to decrease information overload and increases awareness among team members.

1 Introduction

A wide range of personal information management (PIM) tools support knowledge workers in managing their work on desktop computers. These can be broadly categorized based on the main practices they support: the management of tasks, windows, and files. Unfortunately these tools aren't well integrated, causing conflicts where functionalities overlap [8]. For example, window managers support multitasking but are disconnected from the tasks specified in task management tools. Most notably, information related to the same project (or activity) is fragmented across several data collections managed by different tools [2]. These problems are exacerbated within distributed and collaborative activities, in which data in addition can be fragmented across multiple devices and collaborators. Users need to decide where to store data, how to transfer it across devices, and how to share it with others [10]. To address these problems there is a need to rethink the traditional computing paradigm, which is rapidly becoming ill-suited to handle the modern day needs of complex knowledge work.

What unifies task, window, and file management is the actual work they support—the user's activities. Therefore, prior *Activity-Based Computing (ABC)*

systems (or activity-centric computing systems) allow aggregating resources, support for communication, and collaboration, within computational representations of activities [6, 7, 8, 14]. These activities can be managed by the user and are well integrated into the operating system, e.g., supporting easy suspension and resumption of activity resources, thus facilitating multitasking. A recent ABC system, Laevo [8], has explored how to better support the full *activity life cycle*: the creation, use, and evolution of activities over time, including archival after their completion. Supporting the full activity life cycle is needed as part of a scalable solution capable of managing ever-increasing numbers of activities, preventing *information overload*. However, Laevo focuses solely on PIM, managing the activities of just one user, and unlike other ABC or groupware systems does not support coordination and collaboration within cooperating teams.

This paper presents co-Laevo, a system extending on the design of Laevo to incorporate support for cooperating teams. Considering the activity life cycle introduced by Laevo, several implications for design in regards to *cooperative activity life cycle management* are formulated. Extending on the original features of Laevo, activity hierarchies are introduced as a way to allow grouping related activities together. This supports the user in defining more granular activities within the context of higher-level activities. To enable this, co-Laevo introduces an activity time line per activity on which containing sub-activities are displayed and managed. Coordination is supported by allowing users to share selected activity time lines with other users. By orchestrating dependent tasks on a shared activity time line, team members can become aware of each other's past, ongoing, and planned activities.

In this paper, we present (i) design implications for an ABC system supporting cooperative activity life cycle management, and (ii) co-Laevo, a cooperative activity-centric system introducing activity hierarchies and shared activity time lines. Within the scope of this paper, we restrict ourselves to reporting on the results of an iterative heuristic design of co-Laevo and a technical discussion of the newly introduced features.

2 Related Work

Prior ABC systems support the user in constructing computational representations of activities which aggregate activity resources [6, 7, 8, 14]. There is a focus on *activity management* as a whole, rather than providing separate support for the management of tasks, windows, and files [8]. This allows offloading common meta-work, like opening required resources when resuming an activity, to the ABC system. Laevo [8] added temporal support for activities, integrating long-term task management within an ABC system by representing personal activities on a time line. For example, a manager can easily revisit notes taken during a meeting a month ago by opening the time line at that point in time, finding the activity he labeled "Progress Meeting", and resuming it. However, unlike other ABC systems [6, 14] which support users to collaborate on activities, Laevo does not support any form of cooperation.

In this paper, we refer to 'cooperation' as conceptualized within the field of Computer-Supported Cooperative Work (CSCW), where it refers to *interdependence in work*: "multiple individuals working together in a conscious way in the same production process or in different but connected production processes" [11]. Computer systems in support of cooperation are often called 'groupware'. Such systems are generally designed to support a subset of cooperative work, e.g., messaging systems, multi-user editors, or group decision support systems [3]. In addition to the actual cooperative work, *articulation work* needs to occur in order to enable effective cooperation: the coordinating, scheduling, meshing, and integrating of interdependent activities. To this end, *coordination mechanisms* are often put in place [12]. Our work is situated in providing computational support for such coordination mechanisms as part of everyday work in a desktop work environment for knowledge workers.

The work presented in this paper differs from prior activity-centric systems in that it is the first ABC system to support scheduling of long-term cooperative activities in time. This work differs from traditional groupware and coordination tools like electronic calendars or Gantt charts in that coordination mechanisms are supported as part of an ABC system well-integrated into the operating system, rather than being an independent tool. Within co-Laevo the actual resources needed to start work on activities are not disconnected from the activity description (name, representation and schedule), i.e., the actual work is not detached from the plan. In essence users work 'within' a shared schedule, suspending and resuming activity workspaces in order to access required resources. Users are thus constantly confronted with their own and collaborators' activities as part of everyday work when switching between tasks.

3 The Cooperative Activity Life Cycle

As part of the design of Laevo the *activity life cycle* was introduced [8]: a model representing the fundamental practices that influence the state of activities over time (activity construction, interruption, resumption, and closure), framed within three fundamental processes of knowledge work (archiving, multitasking, and planning). In this section, we discuss the implications of the activity life cycle in a cooperative environment where activities are shared. This gives rise to several design implications for *cooperative activity life cycle* management, which we will discuss from three different perspectives (the user, their data, and the different views on that data), summarized in Figure 1.

3.1 Users: Shared Activities

In addition to personal activities, users need to have access to the shared activities of users they cooperate with. This implies that for each activity there is a local, as well as possibly shared context associated to it. Common to both, however, is the activity signifier [5]: a necessary description used to refer to and discuss the activity. To this end, each participant needs to be able to

have a personal workspace associated to a shared activity signifier. Although the workspace is local, it can be used to access shared resources and initiate collaboration with users working on the same activity.

Activity *construction* is "the practice of defining the context of an ongoing or planned activity. During this practice, users gradually build up and modify the content, thus refining the scope of the activity" [8]. Different from PIM, activity construction is no longer restricted to one individual user. As posited by Schmidt and Bannon [11], within cooperating teams there is a need to "support the ongoing dynamic articulation of distributed activities and the cooperative management of the mechanisms of interaction themselves". Activity coordination can be supported by collectively managing the state of activities in a shared work environment (e.g., an activity that changes from a planned state to open, indicating work has started on it). However, in order to keep users that depend on an activity up to date, there is a need to inform everyone within the shared work environment of modifications (these might be possible *interruptions* during ongoing work), including life cycle state changes (e.g., an activity which is completed by another user). In addition to facilitating coordination, this supports team *awareness* by regularly confronting users with the activity descriptions and states of other users while switching between their own activities as part of everyday knowledge work (*multitasking*).

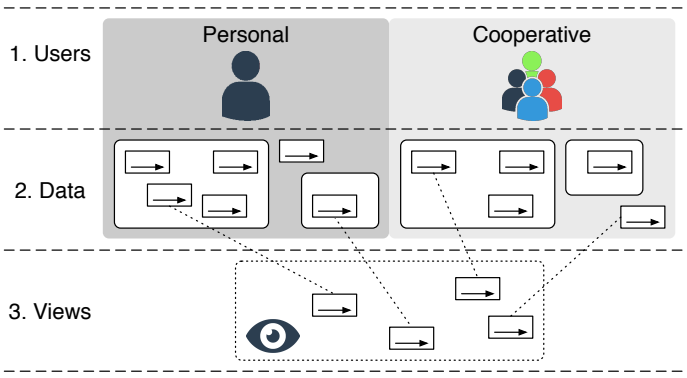

Figure 1: We discuss design implications for a system supporting cooperative activity life cycle management from three different perspectives: the *user*, their *data*, and different *views* on that data. Activities can be both personal and cooperative, thus data structures need to be devised which can support creating ensembles of the two. To prevent information overload (given the increase in the number of activities to be managed) we suggest a seamless transition between a conceptually coherent personal and cooperative view on available activities.

3.2 Data: Organizing Activities

Considering that the number of activities that need to be managed in a cooperative environment is the same as during personal information management, but multiplied (approximately) by the number of users one cooperates with, there is a scalability issue that needs to be addressed. Even more so than in PIM, there is a need to be able to group related activities together and relate them to other groups of activities. Merely representing activities along a temporal dimension no longer suffices (even for small activity sets) since the user needs to be able to share groups of activities with other users. These should be separable from one's own private activities. Additionally, groups of activities might be part of an overarching higher-level goal. For example, programming activities can be shared within a software development team, but furthermore are part of higher-level project management of the overseeing company. It should thus be possible to fluently define, relate, and share, collections of activities.

A permanent internet connection, and thus a continuous up-to-date collection of activities, cannot be guaranteed. Therefore, activity state conflicts can arise when users modify the state of a shared activity while offline. Since the personal planning and associated local workspace of users might depend on the state or existence of the activity, concerned users need to be notified and presented with a resolution mechanism once activities are synchronized. For example, two separate users might have opened (and thus indicate having started work on) an activity. A choice needs to be made whether one of the users abandons ongoing work, or whether work up to that point should be aggregated. Alternatively, one of the user continues work on a copy of the activity.

3.3 Views: Personal and Cooperative Workspaces

Laevo was designed with PIM in mind, providing support to easily *suspend and resume* the activities of just one user. Although cooperative activity life cycle management additionally requires access to the activities of users one cooperates with, not all shared activities should be made part of the personal workspace. This would rapidly lead to information overload. Users thus need to be able to claim ownership (which can be shared with other users) over the activities they are interested in. To further support coordination, it should also be possible to suggest ownership to others. A similar view to that of Laevo (a personalized overview) can show all activities one has *claimed ownership* over, distinct from a view which provides access to the full set of activities one *has access* to. A seamless transition between the two supports frequent switching between them as part of creating and selecting new activities to work on.

4 Design of co-Laevo

The original design of Laevo [8] has two distinct work environments, one *activity management* environment dedicated to creating and managing the state of activities (the activity time line), and several *dedicated workspaces* (one per

activity) dedicated to performing actual knowledge work. On the activity time line, Laevo employs the inherent state changes of activities over time as a way of organizing and accessing them. As demonstrated in Figure 2, activities can be *open* to represent ongoing work, *archived* when completed in the past, or *scheduled* as either planned events at a particular point in time or as to-do items when no point in time to work on them has been decided yet. Activities are displayed along a time line to emphasize their fragmented and parallel execution over time, supporting multitasking [4]. This is distinctly different from the ordinary day, week, or month view in traditional calendar systems. It is up to the user to change the state of activities as they unfold, e.g., a to-do item can be dragged from the to-do list to the time line in order to plan it at that point in time. A *dedicated workspace* per activity, implemented as virtual desktops, provides access to activity resources and serves as a focused work environment free from distractions from other activities.

co-Laevo extends on the design of Laevo by introducing *activity hierarchies* and *shared activity time lines*, which we will describe in this section. Findings from prior versions of the system are taken into account, including the suggested improvement to visualize activity revisitations as 'multiple instances' of the same activity over time, rather than long-running uninterrupted activities [8] (e.g., the "Read Related Work" activity in Figure 3). During the design process a strong emphasis was placed on finding a satisfactory compromise between the original requirements derived from PIM literature (the personal overview of activities as presented in Laevo) and incorporating access to even larger numbers of activities part of a cooperative work environment. The newly introduced features do not

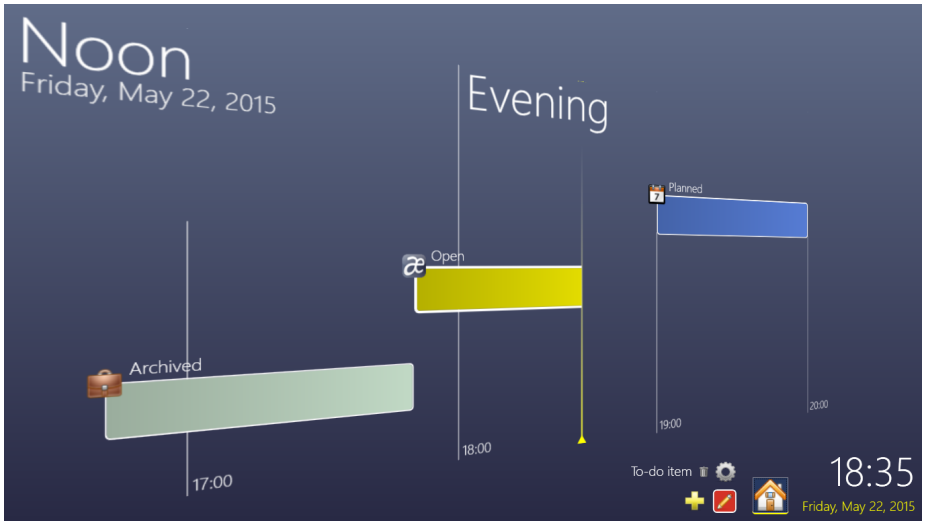

Figure 2: Different supported activity states represented on the activity time line of Laevo: open, archived, and scheduled (planned or to-do).

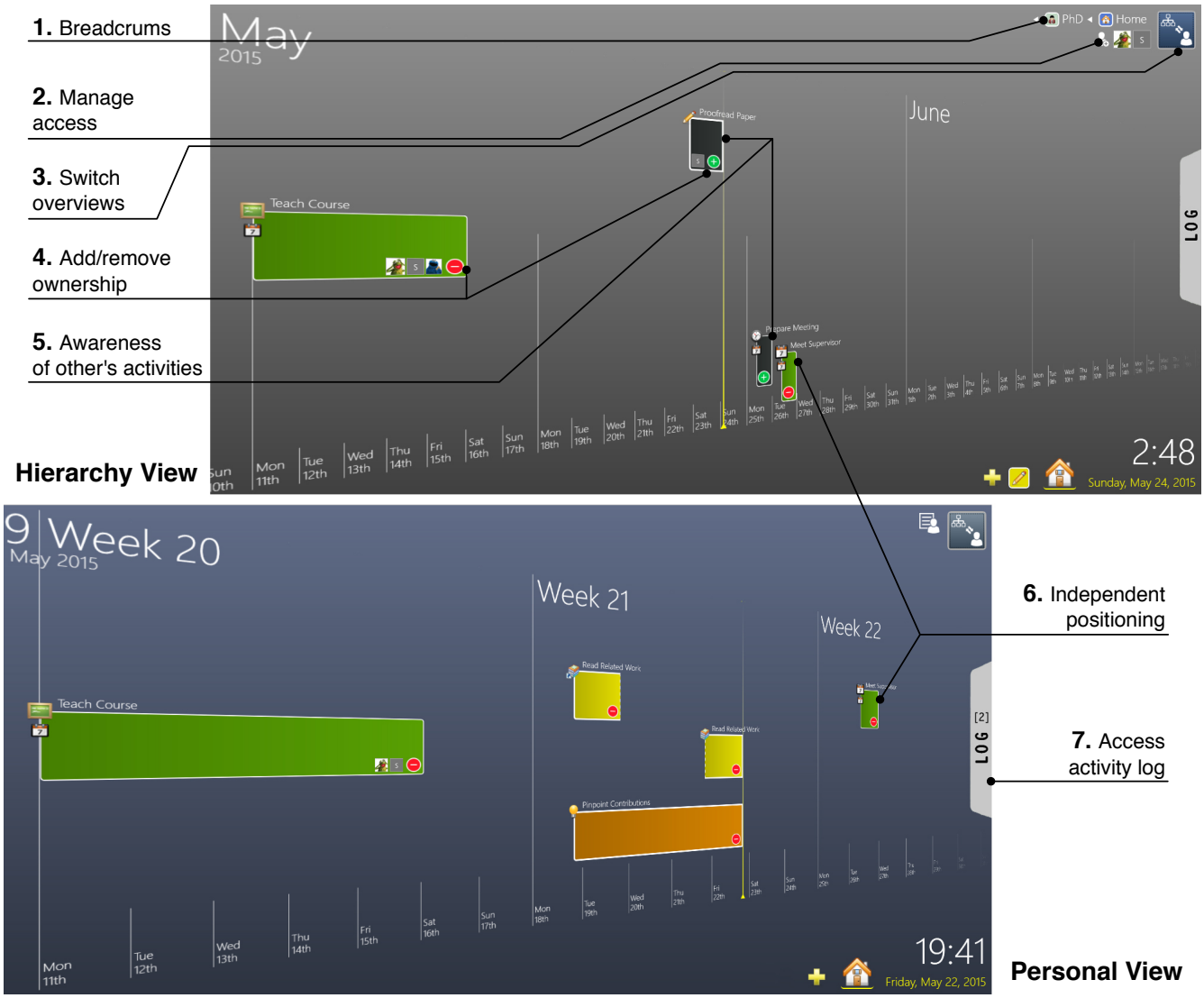

Figure 3: An overview of the newly introduced cooperative features of co-Laevo in both the new hierarchy view, and preexisting personal view.

conflict with the earlier design of Laevo. In fact, co-Laevo can be seen as a conceptually coherent superset of Laevo which merely introduces an additional layer of abstraction to scale up the original design to a cooperative context.

4.1 Activity Hierarchies

As part of the evaluation of Laevo, we noticed insufficient support for high-level activities that need to be revisited only sporadically, like long-term projects representing particular clients. Users expressed problems in managing them: stopping and reopening projects did not feel adequate as the project was not discontinued, but rather, did not require any attention at the time. This indicates a need for the user to be able to define more granular activities within the context of higher-level activities. For example, a meeting activity could be considered part of an overarching project activity. This observation is reflected in activity theory [9], in which activities constitute *actions* directed at specific *goals* which in turn are part of attaining an underlying *motive*. Goals can be decomposed into sub-goals, sub-sub-goals, and so forth. For example, a PhD student might be motivated to graduate (the activity), for which he needs to write a thesis (the action), containing several different lower-level actions targeted at sub-goals, like reading up on related research. Although activity theory distinguishes between activities and actions, there is no such need to make a distinction in the user interface. It is the user who mentally assigns an intent to an activity upon its creation. We are merely interested in the hierarchical nature of goals, which should be reflected in the user interface in order to support richer management of both high-level and low-level goals.

To support the hierarchical nature of activities, co-Laevo introduces a separate time line for each activity within the system. Each activity represents thus not only a point of access to a dedicated workspace, but also serves as a point of access to a time line from where its sub-activities can be managed. For intelligibility reasons (as not to confuse the user) we chose not to make the state of containing activities depend on that of parent activities, and vice versa. Activities can thus contain sub-activities of any state, e.g., open activities can contain planned activities, and closed activities can contain to-do items. There is not much to be gained from automating possible dependencies between the two, in contrast to the added complexity this would introduce. The 'Home' activity of Laevo is used as the root for the activity tree which represents a top level time line. An example activity hierarchy for a PhD student is shown in Figure 4.

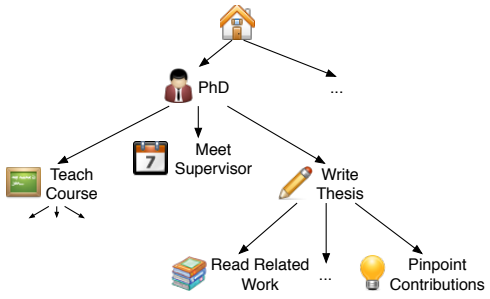

Figure 4: An example activity tree for a PhD student.

The original time line of Laevo provides a complete overview of all the user's personal activities and provides access to their dedicated workspaces. In contrast, the *personal view* in co-Laevo shows only a subset of all accessible activities. In a newly introduced *hierarchy time line view* (top image in Figure 3) all activities in the hierarchy can be accessed and users can decide which activities to show on their personal time line by claiming ownership over them (Figure 3.4). The personal time line thus provides an aggregated overview of a set of selected activities from different levels in the hierarchy view. Browsing hierarchies is similar to how folders are navigated in an ordinary file system, with breadcrumbs indicating the current position within the activity hierarchy (Figure 3.1). The time line shows the containing activities of the currently selected activity (functionally identical to the old personal view from Laevo since there were no hierarchies). The vertical position of activities on the personal view can be modified independently from the vertical position on the hierarchy view (Figure 3.6). This is necessary to maintain a personalized organization of activities in the personal view as opposed to the hierarchy view which can be organized collaboratively. Pressing a button allows seamlessly transitioning between both views (Figure 3.3): the background color changes, the time line is repopulated with the requested activities, but the current visible time interval remains the same.

Compared to Laevo, the personal time line remains largely unchanged, except for newly added cooperative features including a user profile (Figure 5.2), a list of owners per activity, and the ability to remove ownership over an activity (personal view in Figure 3). Removing ownership removes the activity from the personal time line but keeps it in the hierarchy view. Activity ownership is not shown for the active user on the personal time line as this would be a redundant visualization; it is, however, shown in the hierarchy view. Lastly, in both the personal and hierarchy view, an activity log keeps track of changes made to the activities of the currently visible time line (Figure 3.7).

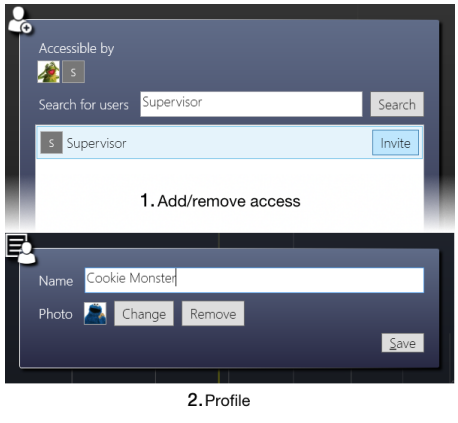

Figure 5: The window used to (1) add and remove access for users, and (2) modify the user profile. Initials are shown when no profile picture is set.

4.2 Shared Activity Time Line

Prior ABC systems [6, 14] provide the means to share activities and their work context with other users and devices. However, support for coordination is limited to sharing *individual* activities. No explicit support is provided to *articulate* (divide, allocate, coordinate, schedule, mesh, interrelate, etc.) activities within cooperative work arrangements [11]. Although some strict interpretations of *situated action* favor highlighting the ad hoc (in situ) nature of knowledge work over elaborate support for planning, the two are not mutually exclusive [13]. For example, activity theory argues that plans are achieved, but undergo continual modifications in the course of action. There is thus a need to support *situated planning*: the plan should be made a malleable part of the activity [1].

co-Laevo supports situated planning by allowing users to access shared activity time lines (Figure 3, Hierarchy View). The notion of *activity access* is distinct from *activity ownership*. Activity access implies having access to an activity and all of its sub-activities (and their sub-sub-activities, etc.), but does not imply activity owernship. Activity ownership means users claim ownership over an activity, at which point it will be displayed on their personal time line. Users who have access to an activity can see who has claimed ownership over it (if sufficient space is available) (Figure 3.4). Users are notified of changes made to activities they own in the activity log of their personal view (Figure 3.7). In short, activity access can thus be used to share and coordinate plans with participating users, and activity ownership can be used to set up a personal work environment.

From the hierarchy view, access can be given to other users to the activity time line currently shown (Figure 3.2 and Figure 5.1). This will trigger an interruption which is added to the to-do list of the recipient, representing the activity they were just invited to. Interruptions which carry context in Laevo are simply highlighted activities in the to-do list. This is distinct from a notification (introduced in co-Laevo), which does not carry context, but refers to an existing activity and gets added to the activity log. Although the initial activity representation (icon, color, and name) corresponds to that of the invited activity, invited users can freely change the representation locally. In essence, only the containing activity time line is shared, allowing users to freely mount the shared activity anywhere within their own personal activity hierarchy using the original activity manipulations available in Laevo. They can thus also choose to represent it as an open, closed or planned activity.

Activity ownership can be suggested to other users, but remains in a pending state until approved. Users are notified of ownership invitations through the activity log on the personal time line (Figure 3.7). In case an owned activity is removed by someone else, the activity is automatically moved to the home time line, removing it from the shared context. Similar to other activity state changes of owned activities, the user is notified of removal through the personal activity log.

5 Technical Implementation

co-Laevo reuses the ABC infrastructure of Laevo [8] to manage local workspaces (including the virtual desktop manager) and introduces a new distributed architecture to manage the sharing of hierarchical activities and distribution of activity related events (e.g., the state of an activity changing from open to planned, or ownership changes). The hierarchical nature of activities is reflected in the distributed architecture, which comprises several peer-to-peer networks each responsible for part of the activity hierarchy, as depicted in Figure 6. When a user modifies a shared activity, the peer-to-peer network responsible for that part of the hierarchy is used to notify all participating nodes. A new peer-to-peer network is created for each branch in the activity tree which includes additional participants with whom the branch needs to be shared. A peer-to-peer network is thus responsible for all underlying activities of the root activity it controls, up to the point where a new peer-to-peer network at a lower branch takes control.

A central peer-to-peer network is dedicated to user discovery (not tied to any activity hierarchies). This network is used to look up currently connected users and invite them to activities, at which point the required information to connect to the peer-to-peer network associated to the activity becomes available. Upon connecting, a full synchronization step ensures the alignment of all participating nodes. Although a peer-to-peer network does not support asynchronous communication between participants, a dedicated seed can easily be introduced by sharing an activity with an always on-line server.

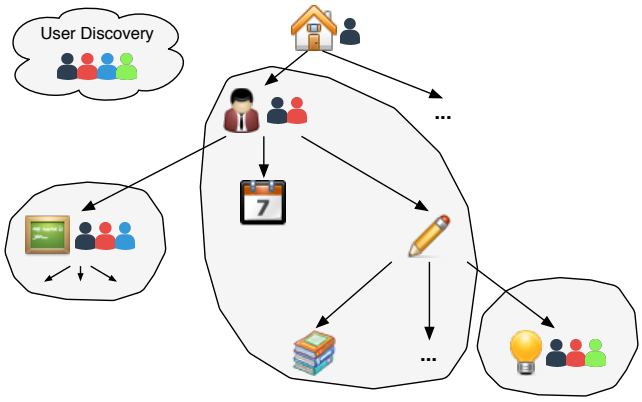

Figure 6: An example of four active peer-to-peer networks for the activity tree represented in Figure 4. One central user discovery network, and three activity peer-to-peer networks, each responsible for a subset of the activity tree. Three separate activity networks are created due to the difference in users whom the branches are shared with.

6 Discussion

To extend on the temporal activity management of Laevo, co-Laevo introduces activity hierarchies. This has two advantages: (i) it allows for richer organizational strategies, since related activities can be grouped within parent activities, and (ii) it simplifies sharing of activity collections since sharing an activity also shares all of its containing activities. By introducing a new user interface to navigate activity hierarchies, very similar to the personal activity time line, the user is supported in navigating both the personal and the full set of accessible activities in a consistent way. Cooperative activity life cycle management is thus supported by allowing *to group related activities together* and by providing a seamless transition between a *personal* and *cooperative work environment*.

To support multiple participants, co-Laevo associates each activity with a *shared time line* and *a local dedicated workspace*. Within the local workspace users can set up their own private work environment, yet access shared resources associated to the activity. As in an ordinary desktop environment, collaboration can be initiated using traditional collaboration tools. Within the shared time line (accessible from the activity hierarchy view), sub-activities of the parent activity can be coordinated, supporting planning and division of labor. Since users need to access the hierarchy view to create or search for new activities to work on, they are regularly confronted with the ongoing, past, and future work of all participants, thus improving awareness within cooperating teams.

7 Conclusion

In this paper, we provided insights for other systems on how to provide support for cooperative life cycle management. In particular, we presented how the interaction technique of defining *ownership* over activities can be used both to filter personal activities as well as to provide coordination and awareness within cooperating teams. As a technical contribution, we described how local dedicated activity workspaces, combined with a distributed hierarchical peer-to-peer network, can be used to support the cooperative activity life cycle. co-Laevo provides explicit support for the articulation of activities by providing access to shared activity time lines from where collaborative activity workspaces can be coordinated and instantiated. Our final design is based on activity theory, theories on cooperation, and insights gained during the iterative heuristic design of co-Laevo. An in-field evaluation of the system is outside of the scope of this paper and will be the focus of future work.

8 Acknowledgments

This research has been funded by the Danish Agency for Science, Technology and Innovation under the project “Next Generation Technology for Global Software Development”, #10-092313.

References

[1] Jakob E. Bardram. 1997. Plans as situated action: an activity theory approach to workflow systems. In *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work*. Springer, 17–32.

[2] Ofer Bergman, Ruth Beyth-Marom, and Rafi Nachmias. 2006. The Project Fragmentation Problem in Personal Information Management. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM, New York, NY, USA, 271–274. DOI: http://dx.doi.org/10.1145/1124772.1124813

[3] Clarence A Ellis, Simon J Gibbs, and Gail Rein. 1991. Groupware: some issues and experiences. *Commun. ACM* 34, 1 (1991), 39–58.

[4] Victor M. González and Gloria Mark. 2004. "Constant, Constant, Multitasking Craziness": Managing Multiple Working Spheres. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 113–120. DOI:http://dx.doi.org/10.1145/985692.985707

[5] Steven Houben. 2014. *An Activity-Centric Approach to Configuration Work in Distributed Interaction*. Ph.D. Dissertation. IT University of Copenhagen.

[6] Steven Houben, Jakob E. Bardram, Jo Vermeulen, Kris Luyten, and Karin Coninx. 2013. Activity-centric Support for Ad Hoc Knowledge Work: A Case Study of Co-activity Manager. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2263–2272. DOI:http://dx.doi.org/10.1145/2470654.2481312

[7] Steven Houben, Paolo Tell, and Jakob E. Bardram. 2014. ActivitySpace: Managing Device Ecologies in an Activity-Centric Configuration Space. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces (ITS '14)*. ACM, New York, NY, USA, 119–128. DOI:http://dx.doi.org/10.1145/2669485.2669493

[8] Steven Jeuris, Steven Houben, and Jakob E. Bardram. 2014. Laevo: A Temporal Desktop Interface for Integrated Knowledge Work. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 679–688. DOI: http://dx.doi.org/10.1145/2642918.2647391

[9] Victor Kaptelinin. 2014. Activity Theory. In *The Encyclopedia of Human-Computer Interaction, 2nd Ed.*, Mads Soegaard and Rikke Friis Dam (Eds.). The Interaction Design Foundation, Aarhus, Denmark, Chapter 16.

[10] Stephanie Santosa and Daniel Wigdor. 2013. A Field Study of Multi-device Workflows in Distributed Workspaces. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*. ACM, New York, NY, USA, 63–72. DOI:http://dx.doi.org/10.1145/2493432.2493476

[11] Kjeld Schmidt and Liam Bannon. 1992. Taking CSCW seriously. *Computer Supported Cooperative Work (CSCW)* 1, 1-2 (1992), 7–40.

[12] Kjeld Schmidt and Carla Simone. 1996. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. *Computer Supported Cooperative Work (CSCW)* 5, 2-3 (1996), 155–200.

[13] Kjeld Schmidt and Carla Simone. 2000. Mind the gap! Towards a unified view of CSCW. In *Proceedings of the Fifth International Conference on the Design of Cooperative Systems (COOP '00)*. IOS Press, Amsterdam 2000, 205–221.

[14] Stephen Voida and Elizabeth D. Mynatt. 2009. It Feels Better Than Filing: Everyday Work Experiences in an Activity-based Computing System. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 259–268. DOI:http://dx.doi.org/10.1145/1518701.1518744