



calo

Stardust

final report

mhci project course, summer
july 30, 2007

SRI Team CALO

Yoko Nakano, Will Haines,
Margaret Szeto, Jing Tien, Brian Ellis



Table of Contents

I. Executive summary	3
II. Evaluation of the current CALO system	4
III. Our solution	5
1. Stardust features and design decisions	6
a. General features	6
b. Task pane	14
c. Notification center	24
d. Task viewer	33
e. Schedule pane	35
f. Packs	38
g. CALO suggestions	41
h. Task automation	42
i. Learning log	43
j. Icon well for application access	43
k. Wizard-of-Oz Implementation	45
IV. Future steps	46
V. Appendix	48
1. Research	48
a. Literature review	48
b. Contextual design	56
c. Personae	69
d. Use case analysis	73
2. Ideation and design	74
a. Brainstorming: initial ideas	74
b. Concept validation	78
3. Evaluative user testing	84
a. Think-alouds with paper prototypes	84
b. Think-alouds with wizard of Oz prototype	92
c. Heuristic evaluation	100
4. Specification Sheet	104
VI. Bibliography	117

I. Executive summary

The CALO project (Cognitive Assistant that Learns and Organizes) is intended to permit project managers and other “overburdened knowledge workers” to offload responsibilities that are candidates for automation to an autonomous software agent. The intended user group includes anyone with managerial positions—including CEOs, executives, and deans of academic departments, and their assistants, secretaries and coordinators in various organizations. CALO uses an artificial intelligence developed by SRI to support overburdened knowledge workers through automation, learning and other agent specific actions. The CMU HCI team was recruited to conduct an extensive user research and propose an interface design that better matches the goals and workflows of the target users. Upon examination, the team discovered several problems on the existing interface. It lacked initial user research, was primarily intended to facilitate the work of AI researchers, lacked consistency in user interaction, and did not make the benefits of the AI apparent to the end-users. The research on the target users revealed essential aspects of their work the interface should support: constant interruption, waiting for others, decentralized information and difficulty prioritizing. Additionally, the interface needed to take in consideration the facts AI changed over time, required training and would make mistakes no matter how good to algorithm is. We aimed to create a solution that solved the problems of the existing interface and accommodated the needs of our target users as well as the constraints imposed by the nature of the system.

The CALO *Stardust* sidebar was built based on four guiding principles: to present information that needs to be visible at all times, to give easy access to other information that needs to be visible upon request, to give users ubiquitous control over their workflows, and to embed training in what users already engage in.

We created CALO *Stardust* to present information users need in a form of a docked sidebar. It aims to give users a quick glance of information to help them decide what to do now and the immediate future. The sidebar also serves as a hub of CALO components and users’ native applications, such as emails and calendar, that are essential in communicating to others and organizing their tasks. The *Stardust* sidebar consists of five panes: the notification center, task pane, schedule pane, CALO suggestion pane, and applications access pane. The notification center reminds users of upcoming events, important CALO actions, and a number of important emails waiting to be read. The task pane gathers active, pending and complete tasks that are automatically ordered by priority by CALO’s AI. The schedule pane either presents today’s schedule in a block view or a week’s schedule in an agenda view to help users orient themselves during a day. The CALO suggestion pane presents files and CALO actions relevant to the window focused by users. Finally, the applications access pane contains icons for other CALO *Stardust* components and users’ most frequently used native applications, namely emails and calendar. CALO *Stardust* allows users to access more detailed information of tasks, file associations, training, documentation, and automation easily from the sidebar in a form of stand-alone windows. This aims to address users’ need for accessing information on demand.

To address the system's need for user feedback with minimum disruption to the user's workflow, training is embedded throughout the interface as right-click contextual menus. Additionally, to counteract the highly automated decision makings from the system, such as automatic prioritization, we allow users to modify all data which subsequently trains the system as well.

Finally, to accommodate users who can't afford to lose any screen estate, we designed a minimized version of the sidebar called the mini-bar that still notifies users of important reminders and gives access to necessary information. The mini-bar works essentially the same as the sidebar except that only one pane can be open at a time, and a growl message would come out for important notifications. This keeps users informed and a click-away from their essential information.

II. Evaluation of the current CALO system

The CALO system, as designed by SRI, uses a number of specialized artificial intelligence algorithms to help support DARPA project managers as they organize their work. To support the impressive breadth of this mission, CALO spans several areas of work including scheduling, task management, collaboration, and document management among others. Currently, CALO addresses these topics through a number of separate components, which are individual applications developed relatively independently. For instance, Towel is the to-do manager that helps users organize their tasks. PTIME is an automatic meeting scheduler and negotiator. Mercury or Meeting Assistant records and transcribes conversations to create notes and action items for the user. All these components are valuable applications for the current CALO system, but their lack of integration and consistency makes the CALO system as a whole difficult to use. Furthermore, due to a DARPA-mandated focus on meeting artificial intelligence benchmarks, SRI was unable to extensively research target users and their work styles before designing the system.

Currently, CALO's components are spread out across different applications. This has become problematic because the interface styles and models of interaction in each component differ and sometimes even conflict. For instance, in the application Towel, the user interacts with CALO by manipulating items in a sidebar. However, double clicking on a task brings up a dialogue where the user now has an instant messaging style conversation with CALO in order to delegate a task. Over the course of one task, the interaction technique changes drastically, requiring the user suddenly to interact with an anthropomorphic representation of CALO. Different still is the year three IRIS interface, in which CALO resides in a monolithic browser-like window that explicitly contains the agent's scope. Here, the user interacts with the CALO AI by answering questions to confirm whether or not the system was correct in taking some action. Since the intended model of interaction with CALO changes within components as well as across applications, it is difficult to leverage knowledge from one application to another.

Another problem with the current system is that there is no consistent way to conceptualize CALO. It is not clear whether CALO is part of the operating system, a module of the user's existing applications, or a standalone application. Currently, the user's mental model of CALO is inconsistent as it is presented as an application in PTIME, a browser in IRIS, and an anthropomorphic agent in Towel. We hope to address these issues by standardizing how the AI is presented to the user by using the interaction technique most appropriate to actual user work processes. CALO is already a very powerful and useful implementation of cutting-edge artificial intelligence; now our team needs to make it clear to end users how to harness this impressive power.

III. Our solution

Our interface solution, CALO *Stardust*, presents users with a consistent sidebar application that is powered by an intelligent agent. The *Stardust* sidebar would always be visible on the user's desktop, providing them easy at-a-glance information and quick access to other applications. We designed *Stardust* to leverage some of CALO's current AI capabilities to support users' work in an appropriate manner. Our focus is on assisting target users to decide what to do right now and using the CALO AI to support their immediate task at hand. *Stardust* acts as a hub of many of the user's tools and communications, and it can help them to organize their tasks, schedule, and resources. It observes the actions associated with files, emails, projects, schedule events, and other resources on the computer and learns the patterns of the user's work. Using CALO's powerful AI, we envisioned that *Stardust* would make informed decisions about how to help users manage their tasks, incoming emails, documents, and meetings while continuously improving its decision-making accuracy. It does all of this while presenting a consistent front end to target users so that they always know where to go to interact with CALO. Through *Stardust*, we hope that end users will be able to see that CALO's ability to learn can potentially benefit their productivity.

Stardust is comprised of five panes: the notification center, task pane, schedule pane, CALO suggestions pane, and application access pane. The notification center is the central location for *Stardust* to display alerts that the CALO AI generates for the user based on what it infers from the user's activities. The task pane is a task management system that helps users determine at a glance what to do "right now." It lists the user's tasks by AI-generated priority, and enables the user to open a separate window called the "task viewer" for more detailed task organization capabilities. Similarly, the schedule pane is connected with the user's calendar application and displays the schedule for today and events up to five days into the future, facilitating time-sensitive planning of today's schedule. The CALO suggestions pane is where *Stardust* can display actions and resources that it thinks are associated with the application or document that the user is currently using or editing. Lastly, there is a set of icons on the bottom of the sidebar that facilitates the access to all the important applications and CALO components to which the sidebar is connected.

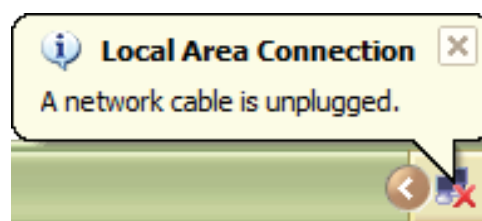
1. *Stardust* features and design decisions

a. General features

Sidebar

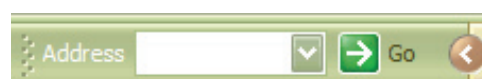
Our contextual inquiries clearly indicate that users interact with two kinds of information in the problem domain in which *Stardust* is intended to exist: information that needs to be visible at all times, and information that should be accessible on demand [V.1.b]. Always visible information is particularly problematic because there is a constant tension between the amount of information to be displayed and the amount of screen space consumed. As such, we considered various alternative interfaces to present the information which needs to be constantly visible on screen. Among the alternatives we considered are a system tray icon, a toolbar, a floating window, a monolithic window, and a sidebar.

A system tray icon has the advantage of consuming almost no screen space. Tray icons can provide notifications to the user by means of pop-up balloons which appear on-the-fly and can be dismissed by the user or clicked for more information. They are incapable of any more complex interaction techniques, however, and any user-initiated interaction must be performed through the use of a contextual pop-up menu invoked by right-clicking the icon, a limited and somewhat esoteric mechanism with very little, if any, affordance. Additionally, the amount of information that can be constantly visible to the user without undue intrusion is limited to that which can be displayed in a 16 x 16 pixel icon, which is to say very, very little.



A Windows system tray icon

Windows toolbars are a somewhat less common interactor. An existing component of the CALO system, CALO Express, uses a toolbar as its sole point of contact with the user interface. Toolbars are more flexible than system tray icons in that they can contain arbitrary controls, and they maintain the ability to be integrated into the Windows taskbar. Their drawbacks include the requirement that they not be taller than the taskbar when it is on the top or bottom of the screen, nor wider than the taskbar when it is on the left or right, making their overall size both limited and unpredictable. As such, a toolbar presents only slightly more opportunity than a system tray icon to present always-visible information, and is primarily useful for the function it serves in CALO Express: as a means of input rather than output.

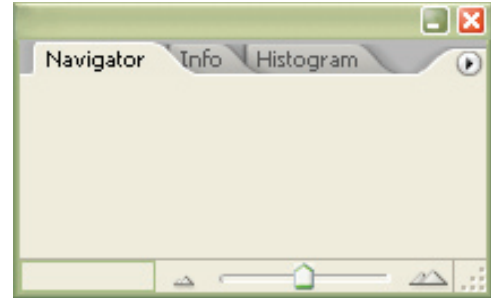


A Windows toolbar

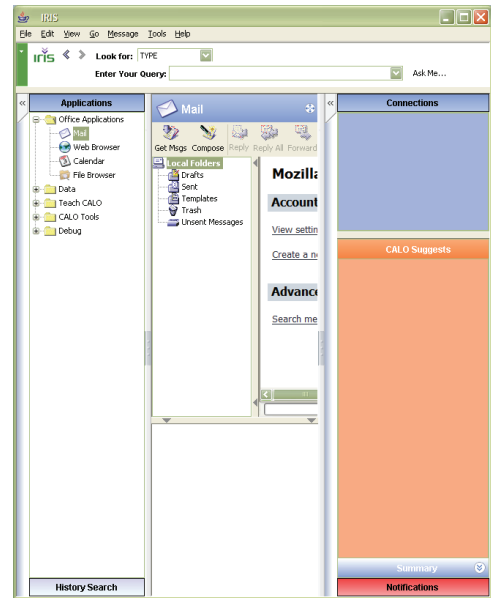
Floating windows, or palettes, are simply normal windows with the special property of remaining on top of all other windows even when they are not in focus. They can be of any arbitrary size and shape, and are thus much better suited to displaying information, but

their lack of integration with the Windows interface means they tend to occlude other windows and must frequently be moved out of the way to see what is below them.

Monolithic windows are the typical approach of desktop applications designed for focused use for a particular purpose. When the window is in the foreground, possibly even maximized to fill the screen, all information inside it is constantly visible regardless of its importance. When another window is focused, it may cover the original window and render its contents invisible to the user, but this seldom matters because the user is not interacting with the hidden application. Such windows can also present multiple problems in terms of information overload and difficulty of finding relevant information. The existing CALO interface, IRIS, employs a monolithic window of this sort that contains within it all the applications and information of which CALO is or can ever be aware. Since *Stardust* is not intended to be used primarily in this manner, and the AI is aware of information not contained in any particular window, a monolithic window is clearly inappropriate, a view supported by both previous research and consensus at SRI.



A palette window



The monolithic IRIS window

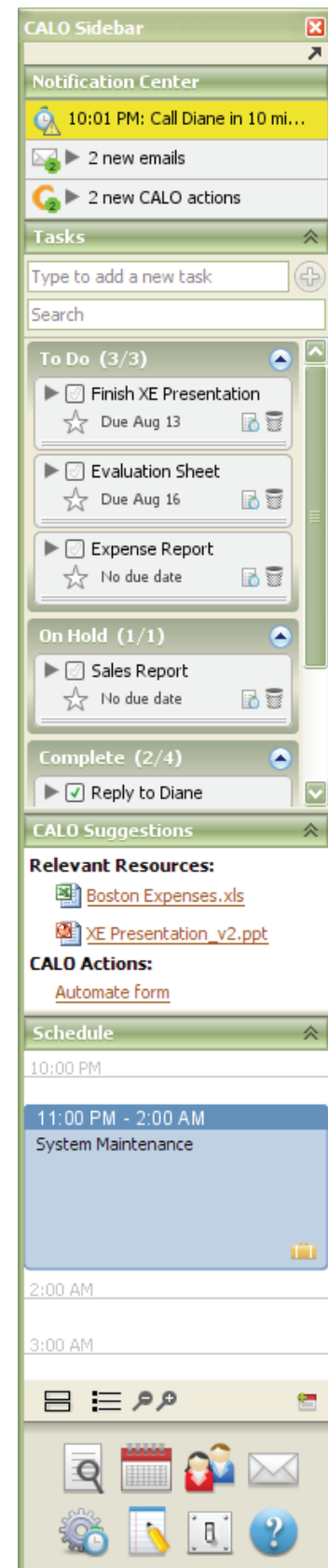
Sidebar have many of the advantages of floating or monolithic windows, in that their size is less constrained than a toolbar or system tray icon's, but they enjoy an integration with the existing interface of the Windows operating system that normal windows lack. In particular, maximizing windows when a sidebar is open will cause them to stop short of the sidebar, thereby ensuring automatically that windows do not overlap or occlude the sidebar. The sidebar takes up an entire edge of the screen from corner to corner, but the length of its other dimension is entirely unconstrained and can range freely from a tiny sliver to a behemoth of a window spanning half the screen. The user is capable of manually resizing the sidebar to fit his or her preference. Therefore, we agreed upon a docked sidebar as the best solution for *Stardust*'s interface, so long as we provide a way for users to flexibly adjust its size to meet their need for available screen real estate. As for the location of the sidebar, the right side seemed optimal as it does not cover or displace applications and icons that are on the left side of the desktop on Windows, nor does it interfere with the Windows taskbar (itself a sidebar) which is usually placed on the bottom or top edge.

In order to keep the sidebar as narrow as possible, and to preserve the distinction between always-visible and on-demand information, the only information presented in the sidebar

is that which users need to orient themselves within their work. This allows users to focus on the task at hand rather than forcing them to turn their attention to the task of managing other tasks. Our background research, contextual inquiries, and concept validation all indicated to us that the most valuable information for this is that which allows users to understand what their situation is at the present moment and perhaps slightly into the future; high level planning and task organization activities are seldom carried out in the middle of working on something, but are rather integrated into the down time between tasks [V.1.a, V.1.b, V.2.b Fig 19]. In order to get more information or details from the sidebar, users can double-click the panes in the side bar to open up stand-alone windows such as task viewer, calendar, and pack window. The sidebar's purpose is to contain information users need at all times, and given the limited space, it does not contain everything users can do with each pane. The additional information is accessible upon request. For example, to perform advanced filtering and organization of tasks, users must open the task viewer. This concept was later positively confirmed by users in paper prototype think-alouds [V.3.a Fig 31].

Multiple collapsible panes

From our contextual inquiry and concept validation, we gathered insights about which information users would need to see at all times [A 1.b, 2.b]. Some persistent information sources were applications such as the user's calendars and email clients, while others were scattered bits of information, things they might jot down on a piece of paper. Users also wanted access to their current list of tasks, and to be notified of important events. We needed to gather all the requisite information and functionality onto the sidebar while presenting it to users in a useful and accessible form. Thus, we decided to frame the observed clusters of information as separate panes in the sidebar. These panes visually divide concepts in the sidebar, while allowing them to be synchronized behind the scenes. Thus, we apply a visual hierarchy to *Stardust* while simultaneously providing users with increased flexibility in layout of their sidebar.

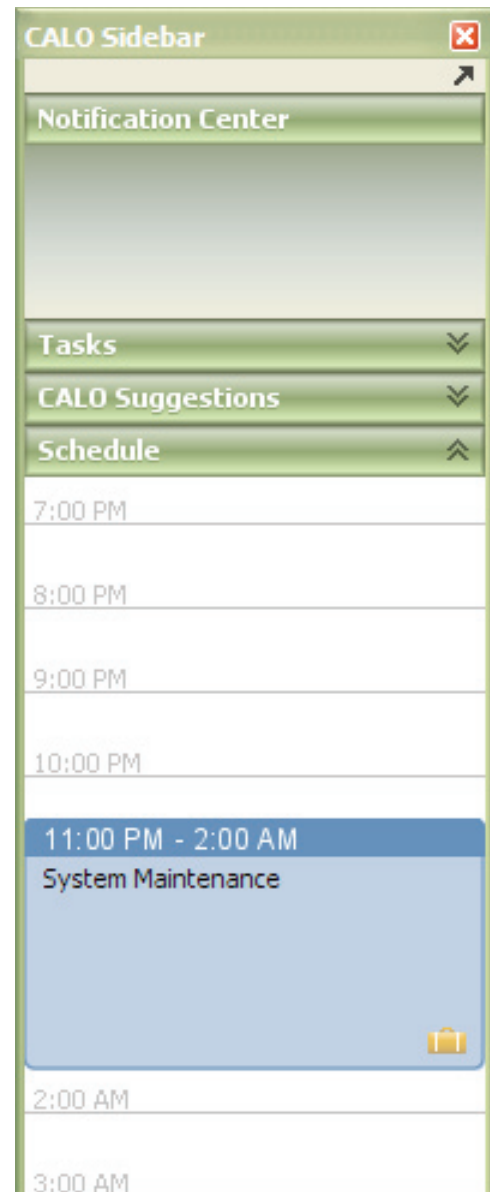


The CALO *Stardust* sidebar

When it comes to customizing layouts, *Stardust* provides users with a fair amount of flexibility to use most of the space on the sidebar to display the panes that they find most useful. Aside from the ability to resize panes by dragging the dividers between them, users can easily collapse or expand most panes in the sidebar. This flexible interface allows users to customize the information that they need to have available on their sidebar; a contextual inquiry documented this user need [V.1.b]. For example, we observed that some people need to see their schedule at all times while others only needed their calendar when planning an event. By allowing collapse and expand, we can ensure that our limited sidebar space is tailored to suit the information that is most salient to any given user. The collapse/expand metaphor extends not only to the panes in the sidebar but also to subpanes in the task pane; making subpanes collapsible allows users to view more of the information they need to see in their open panes. The only panes that are not collapsible in the sidebar are the notification center and access to applications icon well. These must necessarily always exist on screen: the notification center serves as a hub of summary information that users need to be aware of as well as critical reminders, and the icon well serves as a home base to allow users a well-defined way to access CALO components. As such, we disallow resizing these two panes; after all, flexibility is only useful to users if it does not overwhelm the sidebar's ability to show them what they need right now.

Icons and labels

As described above, users of a sidebar have obvious benefits for a system like CALO *Stardust*; however, this interface choice necessarily limits our available horizontal space, restricting our ability to use textual labels. Though it does help to label some obscure icons, labeling everything in the interface would result in unacceptable clutter on a sidebar. As such, *Stardust* uses icons whenever possible to compactly represent concepts on the sidebar. Some of these icons are novel (e.g., the “put on hold” icon) and were initially confusing to users during think-alouds [V.3.b Fig 33]. To address this problem, we have iteratively refined our iconography, and further, we have included tool tip descriptions to help users figure out what an unfamiliar icon might do. Tool tips are included in almost every element of the CALO *Stardust* sidebar to describe various icons, give further instruction about actions, and present more detailed information for various controls and



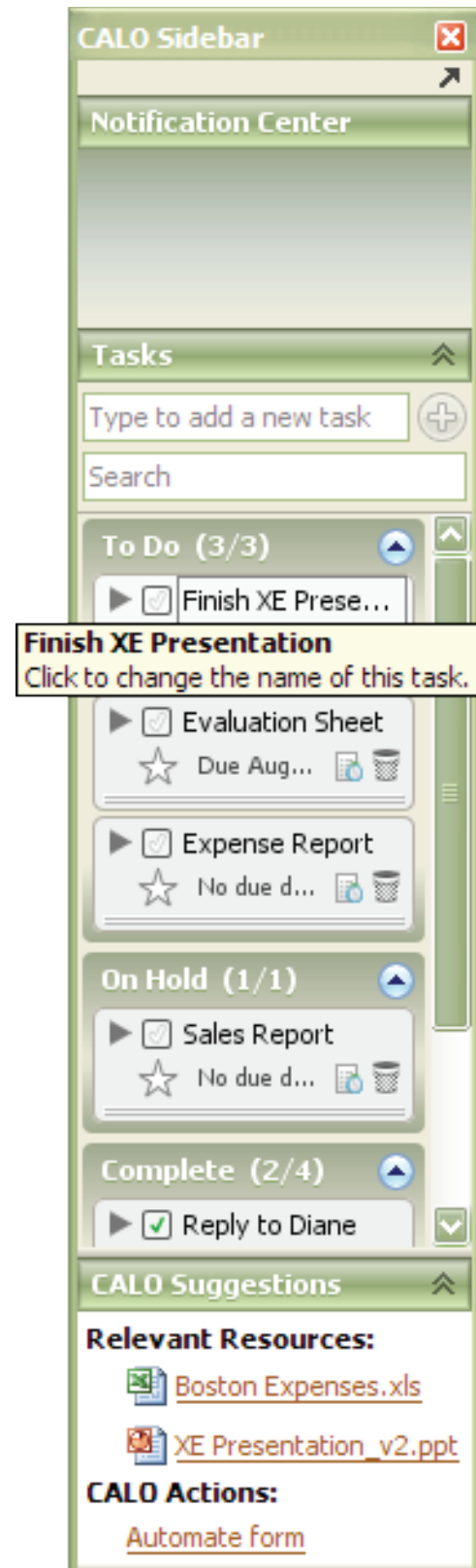
Opened schedule pane and collapsed task and CALO suggestions panes

components. They provide on-demand textual information while not taking up valuable space on the sidebar. We also used tool tips to overcome the problem of the limited horizontal space of the sidebar, which causes lengthy text to become truncated depending on the bar's user-defined width. The tool tips turned out to be extremely useful to users during the implementation think-alouds [V.3.b]. Since the concept of CALO is novel and there are components that are not completely intuitive, many users used the tool tips for quick help. Thus, having a descriptive explanation was crucial.

Mini-bar

We observed during contextual inquiries that many executives only used one monitor, in contrast to their assistants' multiple large monitor screens [V.1.b]. In addition, during our concept validation session, users expressed the importance of screen real estate and reservations about giving up a large chunk of their screen for a persistent application such as *Stardust* [V.2.b Fig 22]. Users expressed that they would be much more enthusiastic about the idea of a sidebar if it could be hidden or minimized. However, simply minimizing the sidebar to a taskbar button or system tray icon prevents the system from displaying multiple notifications (e.g., urgent notifications and important emails) and reduces users' access to content in the other panes such as tasks and schedule. This led us to design a collapsible version of the sidebar called the *Stardust* mini-bar that takes up minimal space while still giving users quick access to necessary information.

The mini-bar is the slimmed down, minimized version of the CALO *Stardust* sidebar with icons representing the task pane, the schedule pane, and CALO suggestions pane. The notification center is shown in an iconic form, and the icons in the application pane are arranged in a single column. The notification center is not reduced to a single icon, unlike other panes, because the notification center presents frequently updated information the system thinks the users need to be aware of; requiring a click or other user action to see this information



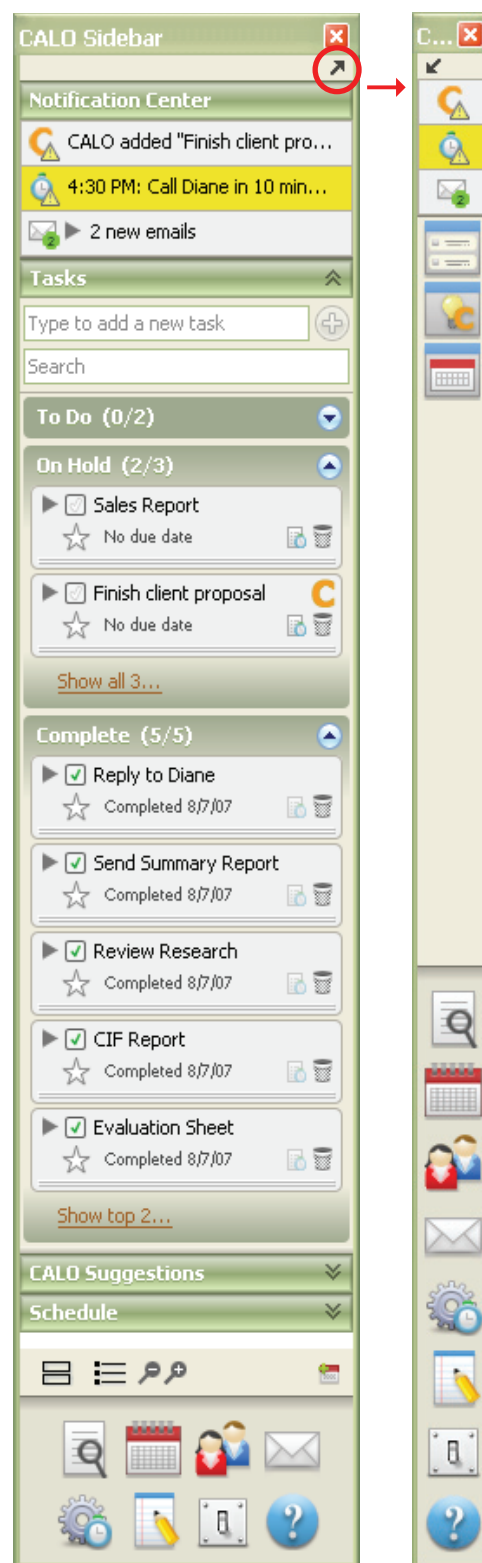
A tool tip appears upon hover

is not sensible. Similarly, we decided to leave the buttons for applications and CALO components available since the mini-bar's layout left plenty of vertical space for them, and it did not make much sense to create a shortcut for shortcuts.

Users can change from sidebar to a mini-bar with one click using a button at the very top of the sidebar. Clicking on one of the icons in the mini-bar will display the pane represented by that icon next to the sidebar, on top of all other windows. Only one pane can be visible at a time, and clicking on the button again collapses the pane. The mini-bar presents a fundamentally different interface paradigm from the full sidebar, since what would otherwise be always-visible information is changed to be on-demand instead. This reduces the usefulness of the CALO suggestions and schedule panes especially, since both of these are designed to be opportunistic: users see something of interest on the pane, and only then make the decision to interact with it. However to compensate for this problem in the CALO suggestions pane, the icon for that pane will light up whenever there is an available CALO action. Since our users consider the ability to minimize or collapse the sidebar to be vitally important, the mini-bar presents a reasonable compromise.

Training

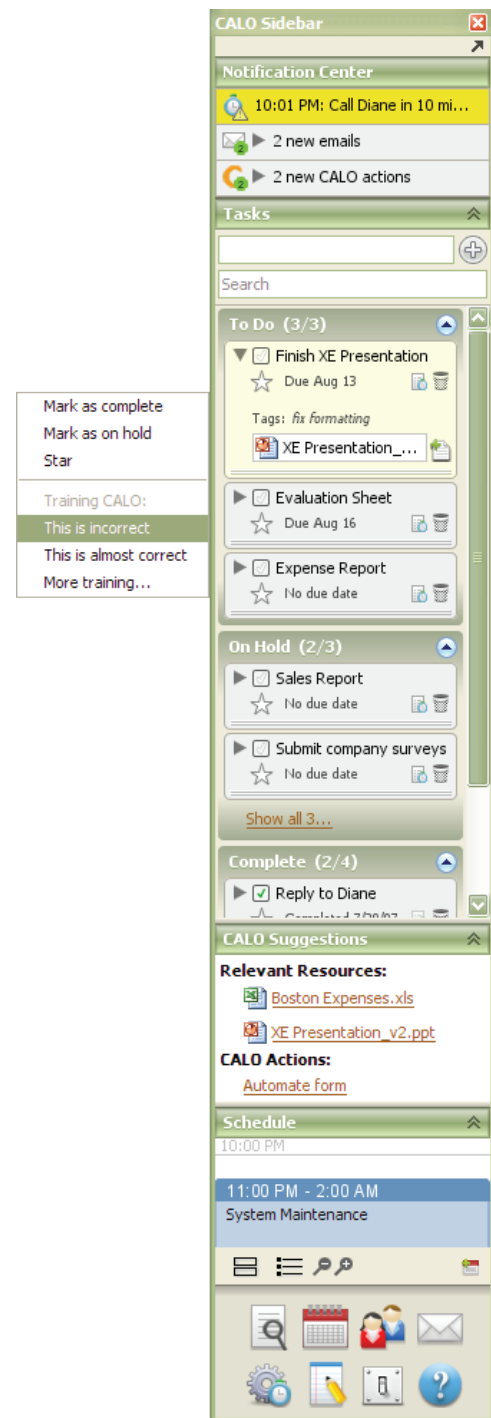
CALO's performance increases dramatically when it receives effective training. Our concept validation, however, revealed that users wish to spend as little effort as possible on training as it tends to disrupt their work flow, consume time, and bring frustrations. This is one of the major problems with the existing system; training is slow, is presented with language users are not familiar with, and the progress of the training cannot be easily visualized or retrieved. Our concept validation further solidified the fact that users wanted the system to implicitly learn as much as possible, thus reducing the amount of explicit training users need to give [V.2.b Fig 26]. Explicit training tends to involve some kind of rules-based instruction, which most users felt to be intimidating and difficult to articulate. Thus,



Switching from the CALO *Stardust* sidebar to mini-bar

we opted for training by correction as our default model—to make training implicit by allowing the system to adapt and learn from users’ actions rather than making users tell the system how to behave. For instance, we incorporated the use of icons to encourage more training from the users. When users click the trash can icon to delete a task, it teaches the AI that the task does not have to be done anymore. The problem with the implicit training is that it is comparatively slow, and it would take a while for the system to perform in a manner that makes the system’s benefits apparent to users. With this in mind, we sought ways to incorporate some explicit training without breaking users’ work flow.

Our solution is to ubiquitously include explicit training as a part of right-click contextual menus throughout the interface. Through the use of right click menus, teaching the system is simple, ubiquitous and consistent throughout the entire application. If users want to train the system, there are two main options in the right-click menu to guide the AI to the correct behavior. The first is to select “this is incorrect,” which tells the system that the item does not belong. For instance, if CALO added an irrelevant file as a resource associated to a particular task, this action would delete the file and teach the system that it should not have been there. The second option is to select “this is almost correct,” which tells the system that it is partially wrong without the user having to define a rule for the system to adhere to. The right-click menu can have more options depending on what object users are pointing at; for instance, right-clicking on a file will include the option “find similar files” in order to add a document that the AI did not include. Additional training can be reached when users open the the system’s learning log. It lists any learning undertaken by the AI along with the sources of information it used to build its inferences. Users can choose to spend extra time training the system when they want by going to the learning log and correcting any incorrect associations the AI has made.



Training CALO in the contextual menu

Animations

Stardust makes extensive use of animations in its user interface. While at first glance this may seem like a profligate use of superfluous visual effects, animations actually serve an important purpose. In an AI-backed application such as *Stardust*, the system can conceivably make changes to the user interface that are not in response to some user action. Therefore, it is imperative that users be kept aware of such changes to the maximum possible degree. This is not always possible if, for example, users are not at their computers at the time the interface changes. The AI takes care only to initiate changes to the interface soon after user interaction partly to minimize this problem. This still leaves, however, the possibility that users' attention will be elsewhere when the change occurs in the sidebar.

While human perception is such that one will usually attend to a sudden change in one's peripheral vision, the change must be perceived in order for attention to be directed there. The eye is constantly moving from place to place in small, unconscious movements called saccades, which take between 20 and 200 milliseconds [46]. During that time, the visual cortex stops processing incoming visual information, so if a change to the sidebar occurs during a saccade, it will not be perceived by users regardless of its magnitude. This assumes, however, that the screen update takes less than 200 milliseconds to occur. This is why *Stardust* employs animations.

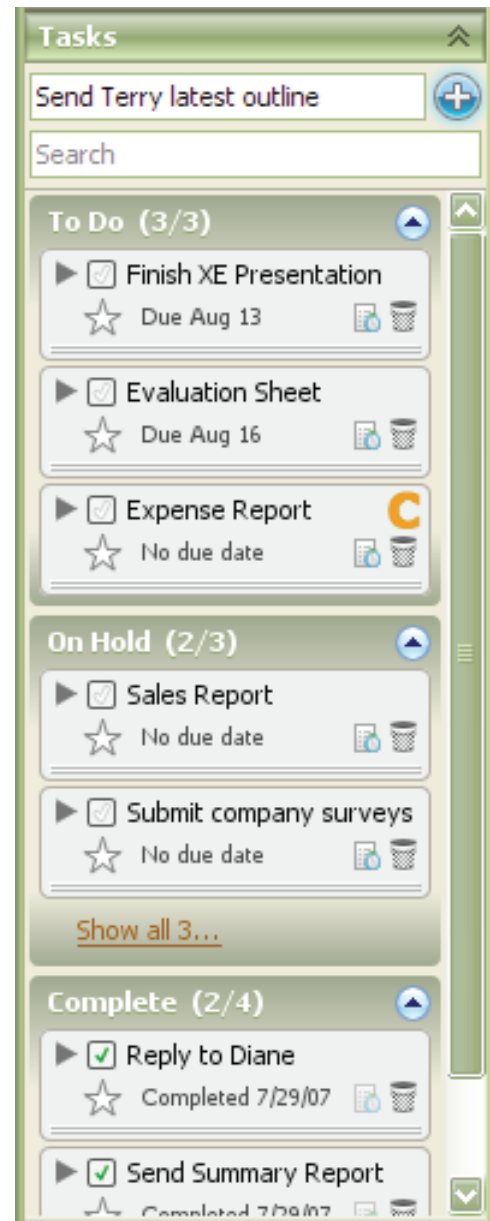
While some animations in *Stardust* do occur in direct response to user actions—collapsing or expanding subpanes in the task pane, for instance—most are initiated when the system makes a change without users' direct interaction. When the AI reorders tasks, the tasks animate to their new locations over the course of at least 750 milliseconds to ensure that users will have an opportunity to perceive the movement. When the AI adds a new task, the task fades in and the other tasks slide out of the way to make room for it, again over 750 milliseconds. The block view in the schedule pane (further details in the “Schedule pane” section) displays users' meetings for the day as “blocks” of time and animates the motion of these blocks over the course of the day (although it is rare for a meeting to move more than one pixel in a single update).

The notification center especially makes frequent use of animations. This is not surprising considering that a major role of the notification center is to capture users' attention when important aspects of the system state change. In particular, when the sidebar is in “mini-bar” mode such that labels for notifications are not visible, high-priority notifications animate out of the sidebar, hovering over the desktop for a few seconds before fluidly disappearing again. New notifications also smoothly animate in, and high-priority notifications exhibit a pulsing effect for the first few seconds to further ensure that the user's attention is drawn, if only briefly, to the text of the notification.

Our team has considered but did not have the time to explore the ways in which users can undo their actions. *Stardust* does not have menus, therefore there is no easy way of showing affordance of the undo function, even with the common keyboard shortcut (ctrl-z) employed. Perhaps the function can be incorporated into the contextual menu, an unconventional but possible location for it to be seen and used universally.

b. Task pane

The task pane is one view of *Stardust*'s to-do manager. It contains three subpanes that hold active tasks, tasks that are “on hold” (a concept described below), and complete tasks, respectively from top to bottom. The tasks themselves are ordered by a priority that is determined automatically by CALO's AI. The subpanes can be collapsed to show only the title bar, normalized to show several tasks, or expanded to show all tasks in the subpane. When the length of visible tasks becomes longer than the height of the task pane, a scrollbar appears. Users can manually enter tasks through a text field above the list of tasks, or the AI may add tasks automatically, such as after parsing the content of emails. When the AI adds tasks automatically, it takes care to wait until the causal relationship between the source of the task and the task itself is clear: for example, if CALO parses an incoming email and determines that it contains a task, it would not immediately add the task. Instead, it would increase the priority of the “new email” notification for that email in the notification center to encourage users to read it, and only then add the task so that the system's reasoning is apparent to the user. Nonetheless, whenever new tasks are added by the AI, users receive a separate notification of this in the notification center so that they can easily keep track of system actions. When a task is collapsed (its default state), it uses two lines to show the name of the task, its due date, a star field, a completion checkbox to move the task to the Complete subpane, a trash can to delete the task without marking it as complete, and a “put on hold” icon to move the task to the On Hold subpane. Tasks can be expanded to show more details, tags and associated resources. The order of the tasks can be moved manually by dragging and dropping tasks around in the pane. Users can text-search through the tasks using the search field below the added task text field. Users can search through title, due dates, tags and files to find the exact task they are looking for. The design decisions made regarding the task pane are described in more detail below.



The task pane

Ordered by priority

One of the major decisions made at the early stage of ideation phase was that tasks should be prioritized by the system to take away users' cognitive load of figuring out what needs to be done and can be done in some set amount of time. We wanted to avoid simply replicating

many other task managers that already existed. Since our target users deal with a large number of tasks, a simple list of tasks would only marginally help them in prioritization. Ample evidence for this exists in the fact that our contextual inquiries and concept validation showed that although many of our target users had tried electronic to-do lists in the past, they had usually abandoned them and reverted to more flexible paper-based solutions, such as attaching post-it notes to documents (associating resources), creating different to-do lists for different areas of responsibility (tagging), and so on [V.1.b, V.2.b Fig 29].

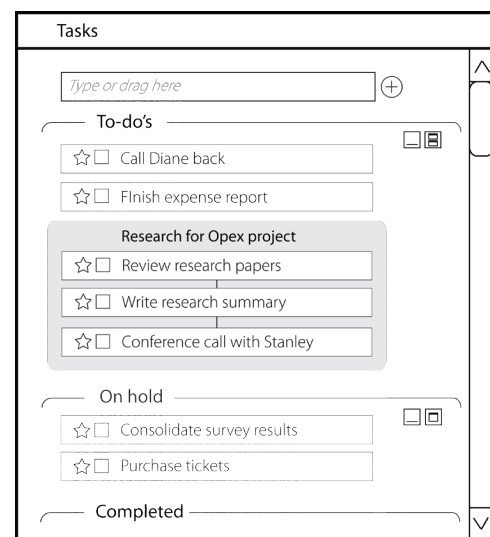
Stardust's conceptualization of priority actually corresponds to two separate concepts as perceived by users: urgency and importance. Urgency depends upon the amount of time left before a task must be completed, and thus changes over time, whereas importance remains fixed for the duration of the task (excepting outside influences). As we have seen in our background research, and especially in work conducted by the RADAR team on task management, these factors are combined implicitly by users when assessing priority: a task that is unimportant may initially be prioritized below a more important but less urgent task, but becomes more highly prioritized as it becomes more urgent as its due date approaches [V.1.a]. The AI therefore estimates both importance and urgency and generates a priority score based on these estimates.

The idea of automatic prioritization was tested extensively at different user study stages, from concept validation to think-alouds [V.2.b, V.3.b]. Some users expressed reservations about using such a feature, stating that they did not want their computers to “tell them what to do.” To enhance users’ perception of being in control of the system, therefore, *Stardust* allows users to modify the order of tasks manually.

Grouping

During our contextual inquiries, we observed that users sometimes have groups of tasks that need to be dealt with sequentially [A 1.b]. These tasks may form the steps of a formal process, like steps in recipe, or each task may simply require that the task before it be complete before it can be started. In the context of the AI system, such groups serve to ensure that the AI does not reprioritize the tasks into an incorrect order or separate them from each other. To support this, *Stardust* allows users to group tasks and create task groups in the task pane by dragging individual tasks on top of each other.

Task groups act somewhat like individual tasks, with some key differences. Although the AI may reprioritize the group as a whole, it will never change the order of tasks within the group (although users can reorder tasks within a group via drag and drop), nor will it split up the group by reordering an outside task into the middle of it. Individual tasks in a task group cannot be



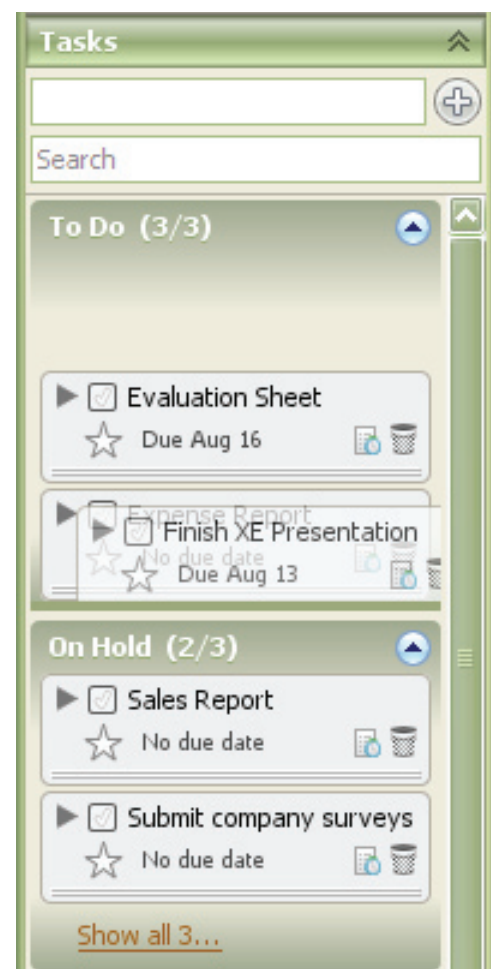
(Wireframe) Grouping tasks in the task pane

marked as on hold; the entire group's status changes as a single unit. Likewise, although each individual task can be marked as complete independently, the task group will not move to the complete subpane until every task within it has been completed. The due date of a task group is the earliest due date of any task contained within it. Each individual task retains its own independent list of associated resources. Task groups can be given an optional title to more explicitly identify any process represented by the group.

The complexity of representing and altering task groups prevented us from implementing them in our current prototype, and attempts to test the interaction using paper prototypes was hampered by the lack of drag-and-drop affordance of the paper representation of the tasks [V.3.a Fig. 31]. Users frequently could not determine how to group tasks together [V.3.a]. While improved affordances in later prototypes largely resolved these issues, testing the changes required the use of working on-screen prototypes in which grouping was not implemented, and thus grouping could not be further tested. Additionally, some details of the interaction remain to be addressed (for example, when a task is dragged onto another task, which task should be first in the group?).

Drag and drop reordering

Though one of the strengths of *Stardust* is the AI's ability to figure out priorities of tasks for the users and automatically move them between to-do, pending, and complete, we still needed to keep the ability for users to manually control these aspects to avoid some of the pitfalls of highly autonomous agents we encountered in our background research [V.1.a]. In our design, in addition to dragging tasks into groups, users are able to drag a task and move it to change its relative positioning inside a subpane, and thus change its relative priority. Although not implemented, the user will also be able to drag tasks from one subpane to another. Drag-and-drop allows direct manipulation of tasks, which is far more intuitive than manipulating the numerical ordering of each task. Another benefit is that it is non-modal, avoiding interrupting users with unnecessary dialogue boxes. In addition, drag-and-drop control doesn't use additional space, which is a substantial benefit for a sidebar whose space is inevitably limited. Lastly, drag-and-drop also provides a good opportunity for users to implicitly communicate their intentions to the AI by reordering tasks. The system is even capable of delivering immediate feedback when users drag a task into a new position by, for example, reprioritizing other related tasks as well.



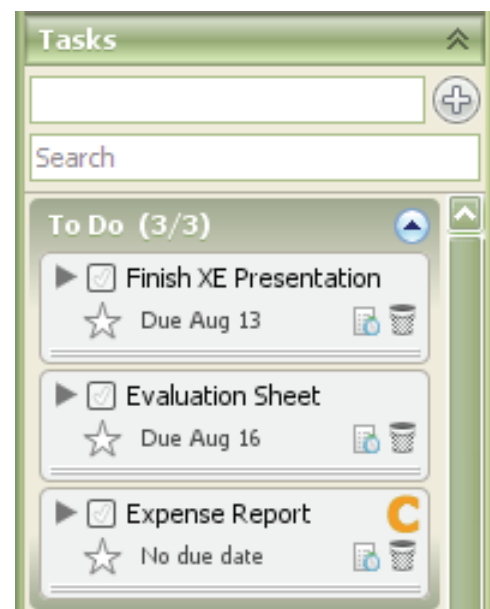
Animating a drag and drop

In our initial think-aloud user tests, we consistently found that the tasks did not afford dragging to the degree necessary for users to discover the feature [V.3.a]. Due to the importance of allowing user reprioritization and the difficulty of providing a different, secondary interaction to accomplish the same goal, we took great care to modify later prototypes to make the interaction more apparent. Knurling was added to the bottom edge of each task, and when the mouse is moved over the draggable area of a task (that is, anywhere not otherwise occupied by a control) the cursor changes to indicate that the task can be moved. Later user tests indicated that the discoverability of this interaction was greatly improved by these changes [V.3.b].

Once users initiate a drag movement, it is important that they know in advance what will happen when they drop the task. This is accomplished by means of a feedforward effect: as users are dragging, if the task is positioned such that it would drop onto another task were the mouse button released at that moment, the task onto which it would be dropped is outlined in a contrasting color different from all the other tasks in the view. If, on the other hand, the task is positioned such that it would be reordered between two tasks, a thick line is drawn in the same contrasting color between the two tasks the dragged task would be dropped between. If the task would be ordered at the very top or very bottom of the list, the line is drawn at the top or bottom respectively. Once the task is dropped between two tasks, it animates into its new position between those tasks.

The “CALO added a task” icon

Another way to mitigate the effect of high system autonomy is to inform users which tasks are being added by the system. Though the notification center informs users of these system actions, checking the validity of each task from the notification center can be tedious. We decided there should be a way for users to quickly glance through tasks to make sure the accuracy of system’s actions. *Stardust* does this by marking the system-added tasks with a CALO icon. Users can look over the task pane whenever it is convenient, verify that CALO has added the tasks appropriately, and click each icon to indicate that the task is correct. If it was not correct, users can also right-click on the task and select an option under “Training CALO” to tell CALO that the task was added incorrectly. This provides highly beneficial feedback to CALO’s AI, as well as allowing users to keep track of which tasks they added manually (tasks which are, at least in theory, guaranteed to be correct and appropriate), tasks which CALO added but which have been subsequently verified (which can also be assumed to be correct and appropriate), and tasks which CALO has added and which have not yet been verified (which may have been added in error). The benefit of distinguishing between the first and second of these states is twofold: first, removing the icon completely would



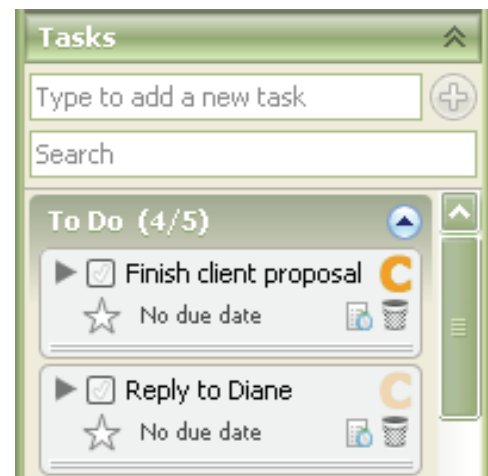
CALO added a task

inhibit user freedom since it would be impossible to “un-click” the icon if it were clicked in error. Second, it increases the visibility of the system state, removing from users the cognitive load of remembering the circumstances in which the task was added. Additionally, since our contextual inquiry observations indicate that many users put only enough information into a to-do list item to effectively cue them to recall all the information they need to complete the task, distinguishing between user-added and CALO-added tasks at all times allows users to incorporate this information into their cue [V.1.b].

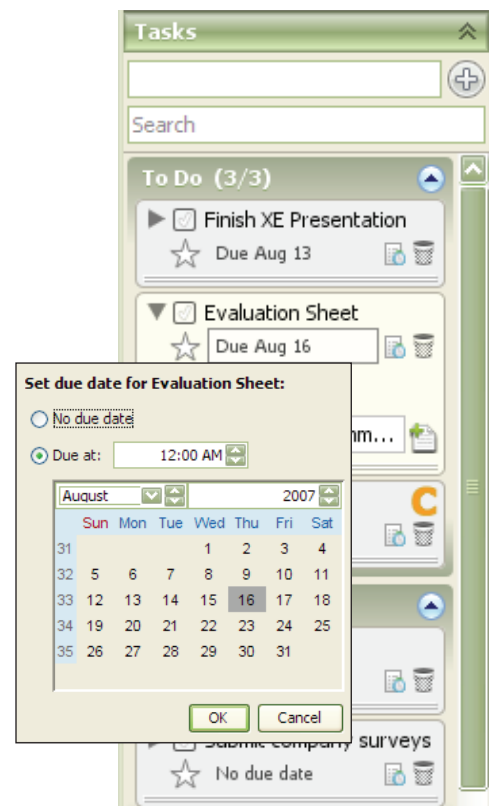
Due dates

We found out during our think-alouds that many users associate priority with due dates [V.3.a, V.3.b]. Since the due date of a task only indicates its urgency, however, and not its importance, simply sorting the task pane by due date provides at best an incomplete model of task priority. Even if all tasks were equally important, simply ordering by due date would result in a random ordering for any tasks that were assigned the same due date or no specific due date at all, and from previous research and our contextual inquiry observations, both situations appear to be a very common situation [V.1.a, V.1.b]. The concept of priority used by *Stardust* goes beyond due dates in that it incorporates other factors such as the project the task belongs to, whether the task is on hold, etc. This is essential for our target users who often have multiple tasks due on the same day.

Our initial interface only listed the names of the tasks when the tasks were collapsed. Because users take due date into account in prioritizing tasks, however, we determined that displaying the due date of tasks in the task pane would provide an opportunity for users to quickly verify that the AI’s task ordering is sensible and provides a greater sense of control. From a design critique, it was suggested that simple task ordering might not be enough information for users to completely verify that CALO’s prioritization is correct. Since the internal representation of priority, a floating-point number, has no intrinsic meaning in users’ conceptual model, we felt it was important to avoid directly exposing this number to users in order to “speak the user’s language,” so we instead chose to list due dates



Users can click on the “C” to tell CALO that it has added a task correctly



Setting a due date

along with task names to help users discern the urgency of each task. Coupled with the user's assumed understanding of the importance of each task, this allows the user to quickly make judgments about the relative priorities of items in the task pane, and to correct them if CALO's AI's judgment differs from that of users.

Multiple groups

During our CIs, our team observed that many users kept stacks of papers on their desks to organize their tasks [V.1.b]. Usually, users keep at least two major piles—a pile for things to get to right now and a pile for things they can't work on immediately because they are waiting for others to provide additional information or actions (a state we refer to as “on hold”) [V.1.b Fig. 3]. The completed task pile is kept separately, often away from their desks. The use of physical piles makes the distinctions between and number of tasks in each group intuitive. Users immediately know how much work is to be done that day, and how much work is waiting for others' responses before they can get to them. The problem with the use of stacks is that the only tasks visible are the ones on the top of the stacks. Though users usually have some idea of where the task might be in the stacks, a computer is obviously better suited to finding information quickly in vast amounts of data. Thus, in our design, we aimed to keep the mental model of stacks while enhancing the search and retrieve actions. The physical stacks of papers were translated to subpanes in the task pane, with each subpane serving as a repository for active, on-hold, or complete tasks. The concept of having a repository for pending tasks was especially well-received by assistants who deal mostly with short sequences of actions that frequently involve waiting for responses from other people [V.2.b Fig. 25]. The Complete subpane would store all the tasks that users or CALO has marked “complete,” and the subpane would empty its repository after a certain amount of time specified by users (one day, three days, one week, etc.).

Filtering of tasks

Even though the sidebar space is limited, we decided to provide a text-search capability for tasks to help target users better manage their tasks. As mentioned above, one of the major advantages of having a digital to-do list is its ability to find the exact task users are looking for in a matter of seconds. Previous research conducted by SRI indicates that overburdened knowledge workers typically manage a large number of tasks (at least sixty), and being able to search and filter through them to work on a particular one is quite important. Furthermore, since tasks tend to shift around as their priorities change in CALO *Stardust*, having a filter to easily search for a task is essential to avoid considerable user frustration. To make the process of searching as effortless as possible, typing characters into the search field immediately filters the task pane to show only those tasks matching the



Filtering tasks

search criterion. This allows users to obtain preliminary search results as they type, which human-computer interaction research (notably that of Ben Schneiderman) has shown to increase the effectiveness of search terms and user satisfaction with the results.

Rather than only allowing tasks to be filtered by title, the search field looks at all textual information associated with the task, including its due date, tags, and associated resources. This allows users to show only tasks due today (by typing “today” into the search field), only tasks due at a certain time, only tasks with a certain tag, or only tasks with which a particular file or email is associated. The search also includes users’ description of the reason tasks are on hold, and the names of people on whom they are waiting, so users can see all tasks waiting on a particular individual by typing their name. In lieu of a more complex query-based search syntax, we feel this provides the greatest flexibility while still remaining accessible and discoverable to novice users.

Our think-aloud user tests showed that the immediate filtering strategy employed by the task pane had one substantial drawback: users tended to become confused when tasks seemed to disappear while using the filter field [V.3.b]. This may be partly due to the fact that the field is labeled “search,” which is ambiguous as to whether it implies a direct filter as seen in applications like iTunes, Windows Vista’s Explorer, and the Mac OS X Finder, or a traditional type-and-hit-enter search as seen in applications like Mozilla Firefox and Microsoft Word. In an attempt to provide better feedback for users initiating searches, a placard appears at the top of the task pane when a search filter is active stating that the view is showing only matching tasks and indicating how many matching and total tasks exist.

Maximize, normalize, and minimize

The CALO *Stardust* sidebar includes several panes, all stacked vertically, so the task pane can only occupy a part of the vertical space on users’ monitors. This means that while vertical space is much less at a premium than horizontal space, it is highly unlikely that users with a typical number of tasks will be able to see all their active, on-hold, and completed tasks at once. In particular, simply providing a scrollbar to allow users to scan down the list would severely reduce the visibility of the “On Hold” subpane. Since contextual inquiry has shown that having on-hold tasks in a separate stack is useful primarily because one’s attention is periodically drawn to them (and thus one is reminded that the tasks are still incomplete), forcing the On Hold subpane out of view largely defeats the purpose of its existence [V.2.b Fig 3].

To avoid this, we provide three states of visibility for subpanes: maximized, normalized and minimized. When a subpane is maximized, all the tasks in that subpane appear in order as one might expect. When minimized, a subpane only shows its title (e.g., “To Do”) labeled with the total number of tasks in the subpane. This allows users to collapse irrelevant subpanes when they only want to see tasks in other subpanes. When a subpane is normalized, users see only the topmost few tasks; since the subpanes are sorted by priority, the visible tasks are the ones that have been prioritized highest by the AI. The number of tasks that are visible in a normalized subpane depends on which subpane it is (more tasks are visible in the To-Do subpane than the Complete subpane, for example) and, in

a complete implementation, would also depend on how much space is allocated to the task pane in the sidebar such that all three subpanes together would fit precisely in the available space without a scrollbar. In the normalized state, the title bar indicates both the number of visible tasks and the total number of tasks in the subpane to clearly distinguish between a normalized subpane that is only showing four tasks and a maximized subpane that only contains four tasks.

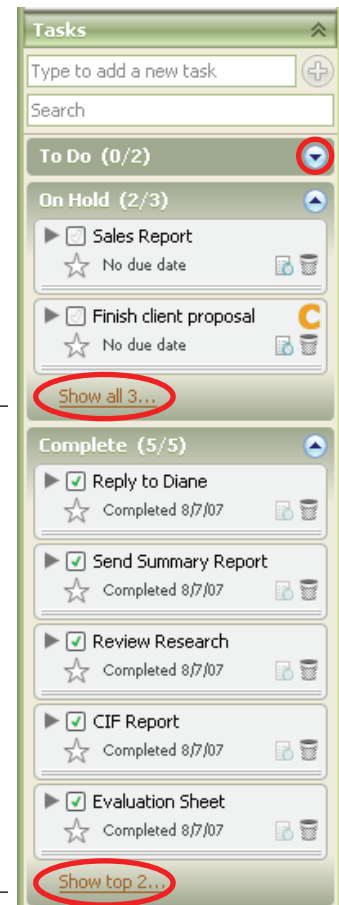
Our initial prototypes featured two buttons at the top of every subpane [V.3.a Fig 31]. The buttons differed in function depending on the current state of that subpane: a normalized subpane would have minimize and maximize buttons, a maximized subpane would have minimize and normalize subpanes, and so on. In order to make apparent the difference between a maximized subpane with four tasks and a normalized subpane only showing four tasks, we added a link-like button at the bottom of each normalized pane for which there are additional unseen tasks “below the fold.” The button performs a maximize operation, and its label indicates the number of tasks that would be made visible by clicking it. When the pane is maximized, the link’s function changes to a normalize operation, and the label tells the user how many items will be visible when the pane is normalized. Users found the buttons at the top of the subpane to be confusing in our think-alouds, in part because the functions of the button at any given position would change once one had been clicked [V.3.a].

They had no trouble, however, with the link, and used it almost exclusively in lieu of the more daunting buttons to maximize and normalize the subpane, only using the button for minimization [V.3.b]. Based on these results, we simplified the interaction model from a three-state toggle (minimized, normalized, maximized) to two two-state toggles (collapsed or not collapsed, and maximized or not maximized). A subpane that is neither collapsed nor maximized is considered to be normalized. A single button at the top of the subpane controls whether the subpane is collapsed, and the link at the bottom controls whether it is expanded. As we later discovered, this matches the interaction of other successful interfaces of this sort, most notably Apple’s Spotlight.

Although subsequent user tests showed the new model to be significantly easier to use, one drawback of the model presented itself: in a maximized subpane containing many tasks, users must scroll past all the tasks in the subpane to reach the link by which the subpane can be normalized.

Minimized
Normalized

Maximized



The subpanes minimized, normalized, and maximized, highlighting the button on the top to collapse, and the links on the bottom of each subpane to normalize or maximize

Scrollbars

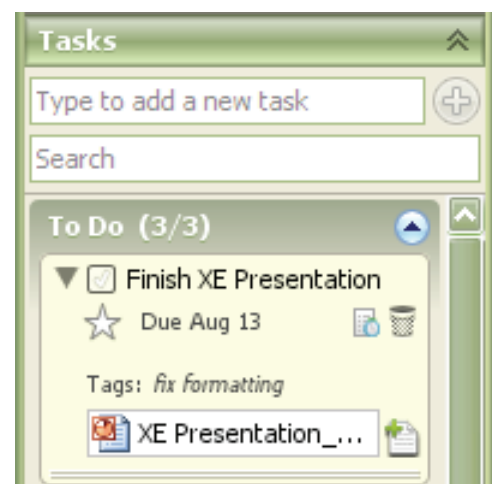
A scrollbar appears in the task pane when the height of the visible tasks becomes larger than the space allocated to the task pane on the sidebar. This is necessary in order to view everything within a limited amount of pane space. The decision was made to have only one scrollbar for the task pane, rather than one for each subpane, so that users would not be obliged to collapse every subpane other than the one they were interested in whenever they wanted to make efficient use of vertical space: if each subpane had its own scrollbar, the already limited space of the task pane would be even further restricted by the height of the subpane that was to be scrolled through, making it difficult for users to visually compare multiple tasks, to say nothing of drag-and-drop. For example, if users wanted to view and interact with only their To Do tasks at the moment, they can simply expand the To Do subpane to let it take up the visual space of the task pane. If they later on wished to view the On Hold and Complete subpanes, they can simply scroll down the entire task pane to see them. This single-scrollbar strategy also eschews the visual clutter of a pane with three scroll bars one on top of the other, and resolves the ambiguity of how large each subpane should be within the task pane. (Should they take up equal space? Should they be sized proportionally to the number of tasks contained within them? Should the To Do subpane always be larger than the Complete subpane? And so on.)

Expandable tasks

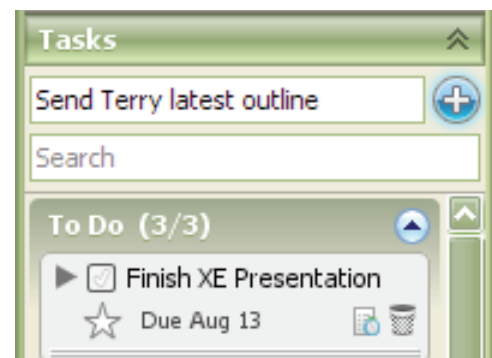
Users can single-click on each task to expand it to view more details such as the associated resources and tags. Since tasks are collapsed by default, this saves some space in the side bar, so users can see more tasks in the limited space. We found that this model is easy to comprehend for virtually all users during think-alouds [V.3.a, V.3.b]. The addition of a disclosure triangle to afford the expansion behavior made the feature easily discoverable by users as well. The decision was made not to limit the number of expanded tasks at any given time, both for consistency with the subpane states and to allow the user to visually compare one expanded task to another. This decision does place the onus on the user to collapse tasks when they no longer need to be expanded, however, or to accept a less efficient use of space.

Ways of adding a task (text or dragging)

We borrowed the way users add a task manually from the existing CALO to-do manager, Towel. We support two ways of adding tasks. The first way is by typing a task name into a text field and either clicking an “add” button or hitting the enter key on the keyboard. The second way is by dragging a resource (a file, email, URL, etc.) into the task pane. This will automatically



An expanded task

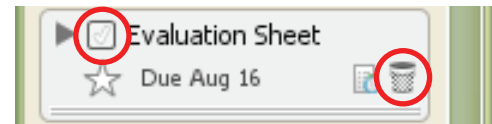


Adding a task

make a task that users can edit later to add more information if they wish. The second option also automatically associates the given resource with the new task, and is especially useful when users want to capture loose items quickly without spending time typing tasks in.

Mark as complete and “delete”

In a typical task manager, the distinction of tasks being completed and tasks that need to be removed from the to-do list for whatever reason (simple cancellation, etc.) is not important. For CALO *Stardust*, there are important differences between these two actions in terms of what and how the AI learns from each. When users mark a task as complete, the AI observes the circumstances in which the task was finished and uses this information to automate task completion for future tasks. When users delete a task, however, the AI must consider the possibility that the task was added by the system in error, or that the need to complete the task has gone away for some external reason (e.g., someone else did it). Having one action that represents both would confuse the system and delay its learning. Therefore, we needed to supply two different actions that represent completion and deletion without either cluttering the interface too much or confusing the users. We used a check box on the left side of each task for users to mark it as complete, and a trash can icon on the right side to delete it.

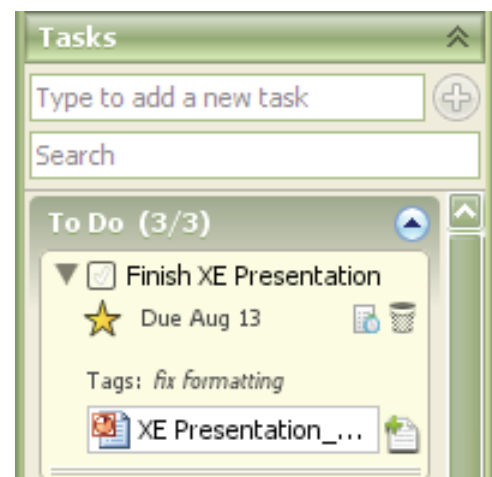


Buttons for marking a task as complete and deleting a task

During think-alouds, we found that many users interpreted the check box as a mechanism for selecting a task, as seen in Web-based interfaces like Yahoo! Mail and Google Docs and Spreadsheets [V.3.a, V.3.b]. Although we initially believed, based partly on the design of CALO's Towel to-do manager, that users would distinguish between Web-based applications such as these and desktop applications like Towel and *Stardust* where selection is usually indicated by clicking directly on the item which highlights it in a different color, this assumption was not validated by the user tests. We therefore modified the “mark as complete” box to use a stylized check mark rather than a native Windows checkbox control, to more clearly indicate the function of the interactor.

Tags and stars

Tags and stars provide additional ways for users to differentiate tasks from each other. They are semantically decoupled, meaning users can use them in any way they want without any strict constraints from the interface. They are also convenient when users want to search or filter their tasks. This convention is also seen in Google applications that a lot of users are familiar with. Users would type in a tag to filter for tasks associated with that tag, and they can also type in “*” to search for starred tasks. In our think-alouds, several users interpreted the star as imparting additional priority to a task [V.3.a, V.3.b]. However, others seemed to approach it as a way to



Tags and star on a task

mark a task as needing attention for later. The various interpretations of the star during the think-aloud suggest that it is serving its intended purpose.

Associated resources

The idea of being able to associate resources with tasks was adapted from the existing CALO. By “resources,” we mean any object that can be represented in the file system. This obviously implies files, but files can be and are used in Windows to represent other forms of data such as Web site URLs and email addresses (.url files) as well as contacts (vCard files), scheduled events (.cal or .ical files), etc. The existing CALO system has a component called “PrepPAK” where users can get associated resources for meetings. Our team expanded on the idea and made the feature available for tasks as well as meetings. The associated resources can be seen when users expand the tasks. The idea was strongly resonated by our target users during the concept validation interviews [V.2.b Fig 20]. It reduces the time for them to search for the file they need to work on and any versions of files they have saved along the way.

While the original PrepPAK idea was preserved in our pack view, described elsewhere in this document, the concept of resource association was modified considerably when it was integrated into the task pane. Unlike a pack, which is generated on-demand and opens in a separate window, the resources associated with a file are always available for immediate view and can be accessed from the sidebar within the task itself. In this, our system more closely resembles the static file associations of the Towel to-do manager, with the notable difference that task pane resources are dynamic and managed by the AI, which can add and remove resources from tasks automatically. This might happen if, for instance, CALO parses an incoming email with an attachment, adds a task for that email, and associates the email, its sender, and the attachment with the task.

Users can also associate resources with tasks manually. There are three ways of accomplishing this from within the task pane interface: first, if users drag one or more resources into the task pane directly, a new task will be created for those resources and they will be associated with it. In the current prototype, the task is given the same name as one of the resources, but one can easily imagine CALO inspecting the resources to determine what the task might be in the same way it can identify tasks within incoming emails. Second, users can drag resources onto an existing task, which will associate the resources with that task. Lastly, because the tasks and task pane do not obviously afford dragging, an “Add resource” button is also provided on each task to allow users to add resources using a standard file chooser dialog.

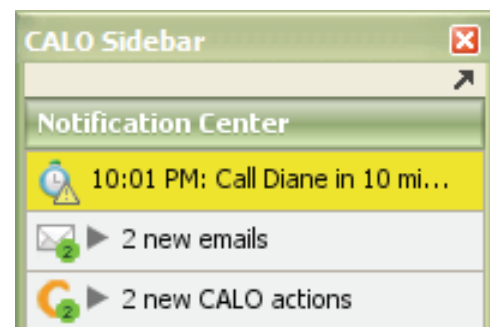
c. Notification center

The *Stardust* sidebar is designed first and foremost to provide users with information that is needed at a glance. However, with limited available space, even persistent sections such as the task pane cannot always display all of their information on screen at once. This is the motivation for the CALO *Stardust* notification center. The notification center serves as

a persistent space from which CALO can place information on the screen in a way that is always visible. The center serves three distinct purposes. First, it is a dedicated space for reminders to alert users of upcoming scheduled events or urgent tasks. Second, the notification center communicates to users about important recent emails, so as to encourage them to check their inbox at the next convenient moment and allow CALO to take action regarding any meetings, tasks, or other information the AI gleaned from reading the emails as they arrived. Finally, it provides a way for the CALO AI to notify users that it has performed an action (for example, added a task, or changed the priority of a task) that they may not have noticed. This ensures that users remain well-informed, even when items change without their direct manipulation.

Three types of notifications

The notification center displays three different types of notifications: timed reminders, important emails, and CALO actions. Each serves a distinct purpose, and is visually separated from the other types using badged icons. The first notification type is timed reminders—these are notifications of time sensitive items which the CALO AI determines would be detrimental should users miss the deadline. Imagine items such as upcoming meetings, teleconferences, or tasks with approaching deadlines; each is “mission critical” for busy executives, and *Stardust* now has the ability to keep these items in users’ visual field where they cannot be forgotten. Further, reminder notifications flash briefly as they appear in the notification center to add a final visual impetus to their appearance. We feel that such reminders are not too intrusive; in our contextual inquiries, we saw several target users already had a notification system of this sort in place, usually one that came with the calendar application they were currently using (such as Microsoft Outlook or Oracle Calendar) [V.1.b Fig 6]. Our think-alouds strengthened this point; without extra visual cues, subjects were apt to ignore reminders while engaged in a task on other parts of the screen [V.3.b].



The three types of notifications

So far the notification center is very similar to systems in place for many of our target users; however, our team recognized that there are other types of notifications that would prove valuable to users other than simple reminders for scheduled events in their calendars. Therefore, we intend the notification center to be the place for all notifications, whether they are coming from users’ calendars, or their task managers, or from CALO itself. A body of HCI literature indicates that emails tend to be the hub of information in the workplace because it is one of the major communication tools employed by users in their work. Our contextual inquiries support this fact; our target users’ work is fragmented with many interruptions and short task sequences, much of which comes in via email [V.1.b Fig 6]. Since email is well known to be overloaded as both a communication tool and an information repository, we noted that users could easily forget items, such as upcoming meetings, or a task that they have planned to do some time ago but left only in email. By gathering such emails together into the notification center, *Stardust* can prompt the user to

read important emails much like a human assistant might vet an executive's inbox. In this way, *Stardust* provides another level of filtering to keep information organized and the mail client less overburdened.

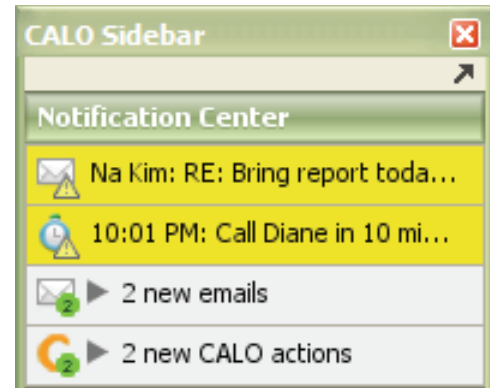
Email notifications provide one added benefit as well. The literature and our contextual inquiries both demonstrate that to-do items arrive in email often and are never transferred anywhere else [V.1.b]. Most email clients are not designed to also function as to-do lists, yet they end up serving that function anyway. The CALO AI is able to parse such emails and add tasks to an actual to-do list, but it may not be obvious to users where the task came from without context. One way to assuage this problem is to provide context to the automated addition by only adding a task parsed from email while users are actually reading that email. This way, they can infer that the task was added by seeing the resource that the AI considered to determine that the task should be added. Such a strategy for showing system reasoning was very popular in our concept validation, but it suffers from a major problem [V.2.b Fig 23]. If users take a long time to get around to reading the email, the task could not get added until it is too late. Email notifications allow the CALO AI to be proactive and ask users to read emails that may contain tasks at their earliest convenience.

The final type of notification is the “CALO action.” These notifications inform users of actions that the CALO AI has taken without direct user intervention. CALO can carry out actions in many different places both on and off the sidebar, and our concept validation strongly indicated that without an ability to supervise, users would be uncomfortable with an artificially intelligent system [V.2.b Fig 27]. To alleviate this discomfort, the notification center unifies important AI activities into one place. For example, a user may be waiting for an expense report to be submitted from a colleague before he or she can balance the company's budget for this quarter. CALO learns from an incoming email that the expense report has just arrived in the user's email inbox, so when the user reads the email it automatically shifts the “Balance budget” task from the On Hold subpane to the To Do subpane in the *Stardust* sidebar's task pane. Upon completing this change, a “CALO action: ‘Balance budget’ task is no longer on hold” notification would appear in the notification center, allowing the user to see the change and be able to easily locate the task when he or she is ready to complete it. Other actions that *Stardust* might notify users about include changing the priority of a task above some threshold, adding or removing tasks or scheduled events, seeking sign-off on some AI-automated task, etc. Our concept validation indicates that the types of AI actions that require notification would vary from user to user and would likely become less stringent over time [V.2.b Fig 21]. Users can learn to trust CALO—keeping them up-to-date is a very important aspect of building that relationship.

Prioritization

As evidenced above by the task pane, a good concept of priority is paramount to *Stardust*'s success at visualizing information in the limited amount of space available to the average user. Notifications are no different, with the caveat that unlike tasks, users have no explicit control over CALO's prioritization of their notifications—they are simply not persistent enough to require such interactivity. There are two broad categories of priority in the notification center: urgent, and extremely urgent. Each notification in the notification

center has its own priority level, and anything above an AI-determined threshold is considered to be an extremely urgent notification. The display order of the notifications is not dictated by the order in which they appeared, but by their priority levels with the highest-priority notifications appearing at the very top of the *Stardust* sidebar. This is consistent with the task pane, and *Stardust*'s overall paradigm for priority; the user always knows where to look to see what needs to be handled right now.



Notifications ordered by priority

Both emails and CALO actions are usually urgent notifications, while reminders, being time-sensitive in nature, are always considered to be extremely urgent notifications. This special treatment of reminders is justified by the fact that they necessarily pertain to users “right now,” and if users do not attend to them, a breakdown may occur. In the case of email, *Stardust* only displays notifications for important new emails. This is necessary because in our contextual inquiries we noticed that many users tend to have a large number of new and unread emails in their inbox at all times [V.1.b]. CALO can determine which emails require notifications by parsing email messages and determining which ones demand users’ immediate attention or affect the tasks that users have to carry out at the present time. CALO actions are similar to emails; again, the AI determines which actions are important enough to warrant sending extra information to the user based on the nature of the action and user preferences. Note that in some cases, emails and CALO actions may also be extremely urgent; there exist emails that require immediate response and AI actions that need immediate user feedback.

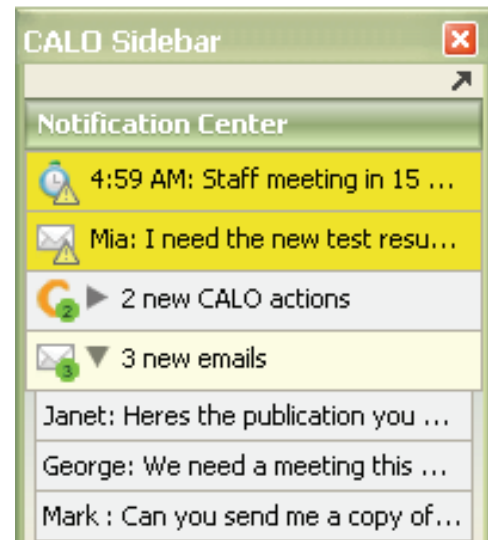
By having a two-tiered concept of priority *Stardust* is able to keep the user well informed with up-to-date, important notifications, while also lessening users’ cognitive load by reminding them of urgent items only when the appropriate time comes. In our findings from contextual inquiries, we saw that our users generally have three categories of “work piles”: tasks with imminent deadlines, tasks that are important but not urgent, and tasks that have far off deadlines or no deadline at all [V.1.b]. The notification center adapts this structure to notifications, allowing *Stardust* to coalesce lower-priority notifications by type into “piles” of which only the topmost is initially visible, as described below. Besides vertical ordering by priority, extremely urgent notifications pulse briefly when added and remain a more vivid color than other notifications, while urgent notifications simply fade in subtly.

Notifications that coalesce and break apart

Most email and CALO action notifications are urgent, but do not need users’ immediate attention; however, extremely urgent notifications could have disastrous consequences if they are not attended to right away. To make the division even more clear and make sure that users do not tune out the extremely urgent notifications, urgent notifications coalesce into groups by their type, while extremely urgent notifications, such as reminders, each appear individually above the coalesced groups. In this way, we preserve the pile metaphor

for urgent notifications, while keeping extremely urgent notifications constantly in users' visual field where they can be read without even moving the mouse. It is as if extremely important notifications each exist on the top of their own pile so that they do not get buried in the clutter of the rest of the screen.

Coalesced groups function more like an inbox, which users check frequently but at their convenience. Instead of having the text of the notifications display their titles, coalesced notifications display information about their contents such as “two new emails,” or “six new CALO actions.” This gives users a sense of how many urgent notifications are building up, while requiring only one extra click to get to the content of the notifications; moreover, it saves a lot of screen real estate in the notification center, allowing more space for other panes. Given the ability to save space and ensure that extremely urgent notification are always the top of the visual hierarchy, coalesced urgent groups seem to represent the best of both worlds.



Coalesced and separate notifications

Associated iconography and badging

In keeping with the general theme of *Stardust*, we visually distinguish between the types of notifications using as little text as possible. This saves space and allows the text to provide only content instead of being overloaded to also provide visual differentiation between notification groups. As such, each notification type has its own unique icon designed to provide enough information to motivate user interaction should a notification be added. Our team was careful to choose our icons for both visual clarity from a design standpoint as well as consistency with the intended Windows platform. We were careful to choose icons based on their meaning in the Windows environment and not overload icons with multiple meanings—actions that are significantly different (for example, delete and dismiss) deserve different icons even if they seem similar to users. This task was complicated further by two considerations. First, some actions have a side effect of training CALO implicitly, and we wanted the icons to indicate what sorts of actions invoke this training. Second, the icons are also intended for use in the *Stardust* mini-bar; here there is not enough space for text, and as notifications come in only the corresponding icon is available to convey information to users. This balancing act was not easy, but we are confident that our several rounds of think-alouds ironed out some initial confusion about icon meanings.

To add necessary information to our icon set without making them prohibitively large, we decided to pursue a badging scheme. While reminders are always considered extremely urgent, there are also situations when emails and CALO actions also fall into this category. When that happens, those notifications appear on the top of the list and are badged with a yellow triangle containing an exclamation point, the universal icon for warning, to indicate their urgency. To stay consistent and append useful information to coalesced

groups, we badge them with a green circle and the numbers of items indicating how many notifications are in the group. This works out well on the mini-bar, where the badge is the only way to indicate that a group has a large number of items in it without requiring direct user interaction. Overall, badging is a helpful way of visually distinguishing otherwise identical icons to communicate the two levels of priority that exists in the notification center. By using badges, the notification type can still be preserved, with urgent or coalesced status appended. Badging is also an easy and economical way of displaying additional information on the *Stardust* mini-bar, where text and space are scarce.

Labels

For the notification to be useful, it must display pertinent information in a very limited space. In general, it needs to provide enough information to prompt user actions within the space of one approximately 200 pixel long line. The problem becomes more pronounced in the *Stardust* mini-bar where there is no space for notification labels at all. As such, our team had to pursue a hierarchical strategy of labeling individual notifications. When email notifications appear in the notification center, the title of the notification displays first the name of the sender, then the email subject, and if there is space left, some of the textual content of the email message. Reminders display a textual description and a time added, and CALO actions only display description text. Each label is backed by a mouse-over tool tip that will display the entirety of the label should it not fit onto the notification. These choices reflect the results of our think-alouds where users expressed the desire to see identifying information for emails, a brief description and time for reminders, and a description for CALO actions [V.3.b].

Our strategy reflects users' current work flows in which emails are labeled by sender and subject, while reminders from programs like Outlook calendar simply show a time and description for schedule items. CALO actions are more novel, but in concept validation, users seemed comfortable with compact descriptions of actions given that there is a way to access more context should users be interested further [V.3.b Fig 23]. By keeping the most important identifying information up front, we are able to cut off the text at the width of the sidebar without losing notification identities; the tool tip reinforces the full message should it be needed. Our users did not have major problems identifying notifications in our think aloud tasks so we are confident that our labeling scheme is sufficient. The fact that it generalizes neatly down to the *Stardust* mini-bar is an added bonus.

Directing users to appropriate places

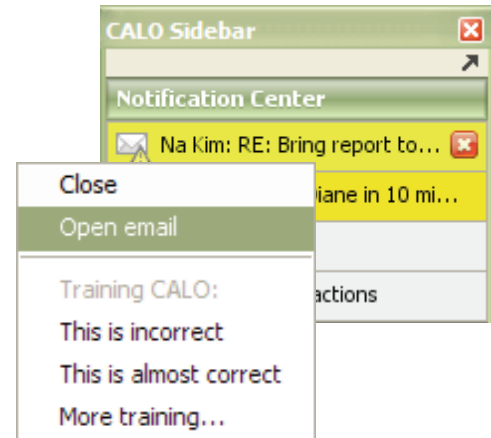
Due to time constraints, our prototype does not implement this functionality, but ideally it would integrate with the user's default email client as well as other applications. The *Stardust* notification center acts as a central location for items that CALO feels users needs to handle in the near-term future. To provide a real benefit to users, it also needs to act as a staging area to get to the locations in which work will actually be performed. In keeping with *Stardust*'s organizational paradigm, users should be able to click through notifications to reach associated items. For example, a single click on a CALO action notification that reads "A new task has been added" would then highlight that specific task in the task pane that it is referring to. A double-click on a notification should open a corresponding

application: for example, a double-click on an email notification would open the user's email client and focus on the associated email.

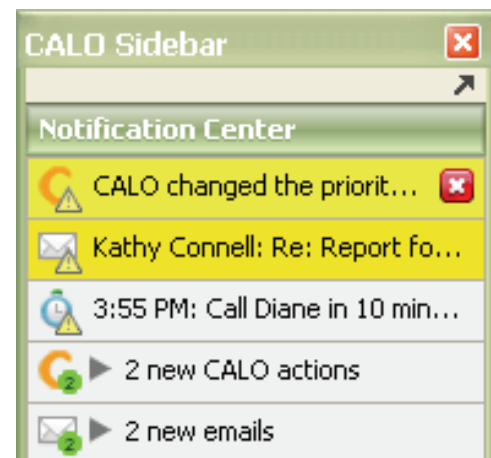
Directing users to associated tasks, resources, or applications provides several benefits. First, by leveraging the notification center as a task staging area, *Stardust* encourages users to consistently refer back to the sidebar when they need information. Users can simply view the notification, then quickly and easily go to the appropriate place where they can either carry out the task or view more details about the notification. Thus CALO becomes more integrated with non-CALO applications and users begin to feel the benefit of having sidebar information available all the time. Further, this convenience can encourage users to carry out CALO-suggested tasks right away when they check the notification, and not delay it for later, requiring a second notification by *Stardust*. Finally, with the user more apt to pay attention to the sidebar, *Stardust* is able to more effectively demonstrate that the system is learning for the users' benefit.

Notification dismissal

Our literature review and contextual inquiries both indicate that the average target user will have a substantial number of urgent notifications at any given time [V.1.a, V.1.b]. Space is at a premium, and if the notification center gets too cluttered, it loses much of its benefit. As such, stale notifications need to somehow be dismissed from the sidebar. *Stardust* uses two strategies to achieve this goal: automatic dismissal and manual dismissal that trains the CALO AI. Notifications automatically disappear after *Stardust* thinks that the user has carried out the action that the notification is informing him or her to do. For example, if a new email notification comes in to prompt the user to check his or her email inbox, the notification would disappear once the user has read the email. In our concept validation and think-alouds, we found that users neither need to see nor want to deal with notifications after the certain task or event has occurred [V.2.b Fig 21, V.3.b]. Usually, in currently existing notification systems, a notification would go away on its own after its purpose is served; users are often just too busy to manually deal with getting rid of a notification. We employ the same metaphor in *Stardust*, but the CALO AI allows us to go several steps further. The sidebar is designed to use its intelligence to support user's progress and constant changes in his or her work flow. Therefore, it can dynamically refresh the notification center at all times as users' circumstances change. Having CALO *Stardust* intelligently carry out actions without user intervention is the unique power behind CALO.



Opening an application from a notification contextual menu



Dismissing a notification with a red delete button

Sometimes, however, the AI will be incorrect or slow to respond to removing a notification. Notifications, by virtue of their transient nature, cannot be richly modified by the user. While it makes sense for users to be able to edit their tasks and their schedule items' details, it would not be useful for them to modify a notification because, after all, notifications exist only to inform the user. Any change that the user makes to a notification would serve no useful purpose—the user has already been notified. Notifications can, however, be dismissed, or if they were incorrect (generated according to invalid assumptions by the CALO AI), the user can bring up the contextual menu for a notification and select either “This is incorrect,” “This is almost correct,” or “More training.” This matches our right-click metaphor for keeping CALO training readily available but just out of the user’s way.

It may seem that users would be unlikely to use this level of training with something as ephemeral as a notification, and our concept validation bears this out [V.2.b Fig 23]. However, notifications are singularly informative learning opportunities for the CALO AI because the notification center is one of the only places where the *Stardust* interface can directly confront the user with items that are the product of CALO’s intelligence and assumptions. In other places in the sidebar (with the exception of the CALO Suggestions pane), *Stardust* indirectly presents its activities to the user under the guise of modifying already extant to-do lists and schedules. Here, CALO implicitly learns slowly from the user’s modifications. However, in the notification center, user feedback on explicit notifications allows the CALO AI to learn much more directly and precisely.

Animation

Much like in the task pane, we use animations in the notification center to demonstrate when CALO performs actions without user intervention as well as to make the sidebar more visually appealing. Further, we leverage animations to focus user’s attention without an unacceptably high level of intrusion. When extremely urgent notifications appear in the notification center, they pulse for three seconds, then remain yellow until the user attends to them. Other urgent notifications would simply fade in to the notification center, and if they were to be coalesced into a group, the coalesced group would simply increase the number of notifications in the group title. From our findings from contextual inquiries, we found that users had many different opinions regarding an appropriate level of intrusion for notifications—some individuals prefer something invasive to motivate them to attend to the matter, whereas other users prefer to have more subtle reminders [V.1.b]. Thus, notifications intrusion levels must be user configurable with a reasonable default setting.

Our team decided to provide non-invasive, but persistent notifications by making them pulse non-abrasively in a



Animating an extremely urgent notification on the mini-bar

steady manner. Making them pulse instead of flash makes them less intrusive. This is very important, as we observed some users who have abandoned notification systems all together because of invasive animations [V.1.a]. Allowing users to set intrusiveness preference settings is an even better option, but as a proof of concept in our think-alouds our middle-of-the-road animation approach appeared to be at least minimally sufficient [V.3.b]. After pulsing, extremely urgent notifications remain yellow to add a bright and solid color in the user's peripheral vision as he or she works, serving to shift user's attention for these immediately important items. This is necessary because highly engaged users may miss the animation if they are attending to other parts of the screen. Other notifications of lower priority do not pulse or have an additional color, serving to act as a contrast and keep the extremely urgent separate from everything else.

Sizing strategy

Even after all of our trouble to reduce the size of the notification center, we realize that there may be more notifications than could be shown in a fixed area. Thus, either the center has to be scrollable or it has to resize. To keep all notifications visually persistent, we chose the second option. The vertical space of the notification center is enough to display three notifications, and if more non-coalescing notifications appear, the notification center pane automatically expands downwards and proportionally reduces the size of the panes underneath it. This all happens without user intervention—in fact, users cannot resize the notification center at all. Our contextual inquiries conclusively indicate that urgent notifications need to be visible at all times, so we do not allow the user to inadvertently make the notification center too small to display everything that it needs to display [V.1.b].

We settled on having the notification center be large enough by default to display three notifications because we expect on average for there to be two coalesced groups (email and CALO actions) and perhaps one extremely urgent reminder. We do not want changes to the notification center to move other panes very often because it is disruptive to user's work. However, space is scarce and we do not wish to make the notification center unnecessarily large. Thus, we decided to make it big enough so that a reasonable number of notifications will fit without the notification center resizing itself, with "reasonable" defined as the number of notification groups that we expect to exist on average. In the case that the notification center does have to resize and push other panes downwards, it would be in a situation where there are a large number of incoming notifications that are of a high priority. In this circumstance, it is acceptable and indeed unavoidable given the other space and visibility considerations to disrupt user's work by resizing the notification center.

The notification center miniaturized

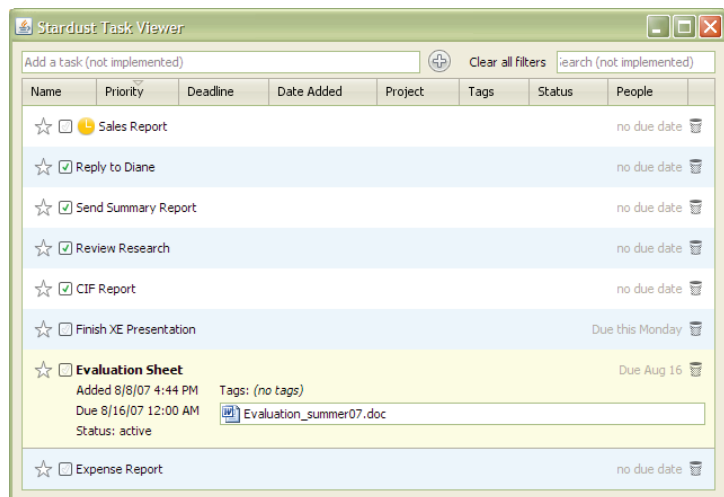
While we can simply hide other *Stardust* panes behind icons in the mini-bar, notifications must be visible no matter how small the sidebar becomes. As such, the mini-bar notification center warranted a full redesign. As described above, mini-bar notification groups are labeled only with icons, dispensing with text labels to save precious space. To get to the full notification, users can simply click the notification icon and the full text notification will appear to the left of the sidebar. This pop up is fully consistent with the full sidebar view, so

users familiar with one version can easily transition to using the other. So as not to lose too much space, only one notification group opens at any given time; opening a new one closes the last view open. Likewise, clicking again on the icon of an open group will close the pop up. Thus, we preserve as much screen real estate as possible without losing any of the functionality of the full sidebar notification center.

Making the notifications openable solves most of our screen real-estate problems, but as we verified in our prototype think-alouds, extremely urgent notifications can easily get lost if they are not made especially visible [V.3.b]. Our pulsing and coloration solutions still apply to the reduced notifications, but we felt that extremely urgent notifications needed even more of a visibility boost. Our solution borrows from the Mac application Growl; when high priority notifications arrive, they are disclosed for a few seconds automatically and then return to the mini-bar by themselves. The animation draws the eye, and the user can read the notification without the extra click required to disclose a notification, dealing with its content right away as necessary. In this way, we can force extremely urgent notifications to be visible while still working within the confines of a tiny sidebar.

d. Task viewer

Our contextual inquiries showed that our target users fall into several distinct demographics with regards to task management strategies [V.1.b]. Some exert minimal control over their to-do list, preferring simply to jot down notes to prompt themselves to remember the important details. Others, however, expend a great deal of effort managing their tasks, adding metadata and investing an inherent semantic meaning in task ordering. For these users, the limited search functionality, space constraints, and above all the enforced ordering by priority in the Task Pane does not permit them the freedom they need. Our concept validation revealed this same divide among our user base, and we found that this latter management strategy was common enough to warrant an interface directly supporting this need [V.2.b].



The task viewer that users can access via *Stardust*

Since the purpose of the sidebar is explicitly to show users information that is relevant to their immediate situation, the sidebar is not an appropriate place for a feature such as this, which is intended for planning and high-level organization of things that may not be directly relevant for quite some time. Therefore, although not implemented in the prototype, the

Task Viewer is intended to appear in a separate window, and be accessible from the task pane and the sidebar's icon well.

Sorting and filtering

Users expressed a desire in our concept validation to exercise control over the order of the tasks in their to-do list [V.2.b Fig 18]. Several mentioned that they needed to see tasks related only to one project, or to sort tasks in order by due date, or to see starred tasks before those that are not. To accommodate these needs, the Task Viewer has rich sorting and filtering capabilities similar to those found in the Explorer application of Windows Vista.

In addition to the “search field” interface also used in the Task Pane on the sidebar, users can see various metadata related to a task (such as its project, tags, due date, etc.) and sort tasks on those metadata by clicking sort buttons (these buttons, which resemble column headers, also appear in Windows Vista's icon view) at the top of the window. As in the Task Pane, tasks can be expanded to allow the user to see more information about them, and associated resources for each task are made available here as well. The resource lists in the task viewer include more associated resources than those in the Task Pane, however, due to the reduced importance of space considerations.

In addition to these sorting features, the Task Viewer employs a more powerful and flexible filtering mechanism, again based on that used in Windows Vista's Explorer. The sort buttons are attached to auxiliary buttons that trigger pull-down menus containing filtering options for the button's metadata type, organized intelligently into groups with a check box beside each. Checking one or more boxes filters the task list to only show those tasks matching the selected criteria. Multiple metadata can be filtered on simultaneously, allowing users to visually build simple filter queries connected with an implied AND operator. A button is provided to quickly clear the filter and show all tasks, and similar options are presented to clear each button's filters individually.

Drag-and-drop

One of the features of the Task Pane in the sidebar is the ability to change the priority of tasks by directly re-ordering them via drag-and-drop. This feature is conspicuous in its absence from the Task Viewer because its semantic meaning depends on the fact that the tasks are always ordered by priority. In the Task Viewer, where the sort order is user-defined, it is unclear what dragging-and-dropping tasks would imply: if the user is sorting by due date, for instance, and the task is dragged between a task that has a due date and a task that does not, to what should the task's due date be set? The user's intention here is functionally impossible for a computer to accurately divide, regardless of the level of AI employed by the system.

Separation of complete and on-hold tasks

Another difference between the Task Pane and the Task Viewer is the way in which complete and on-hold tasks are separated from other tasks. In the Task Pane, there are separate subpanes for each task, but again this forces an ordering on the tasks that the user cannot override. In the Task Viewer, “Status” is provided as a sorting and filtering criterion,

so the user can approximate the ordering of the Task Pane by sorting on Status: on-hold and completed tasks will be sorted below active ones. However, the user is free to override this ordering at any time and see only the tasks they decide are interesting at that moment in the order they think is more appropriate, satisfying the user need of freedom in task visualization and organization.

e. Schedule pane

We decided to incorporate a visualization of the day's schedule into the sidebar due to our contextual inquiry observation that many target users have their calendars constantly open or easily accessible in order to view the upcoming events in their day. In the case that users had multiple monitors, especially for assistants, it was not uncommon for the calendar to occupy one of their monitors. Thus, when designing *Stardust*, the question in our minds was not whether to include a persistent calendar, but instead to decide what visualization of the calendar to present inside the sidebar and what interactions to allow the user to undertake in such limited space.

Scope of schedule pane

Taking into account our limited space and the intended scope of the *Stardust* sidebar, we deliberately limited the schedule pane's scope to be less than that covered by a traditional calendar. The entire pane only shows details for one day or a less detailed view for up five days into the future (not counting weekends). The schedule pane does not show information for more than a few days from now because, in keeping with the purpose of the sidebar, it is not intended for planning. It is, instead, meant to help users orient themselves within the day or week, and inform them of their scheduled events within that time frame. Target users invariably already have a calendar system that they use for keeping track of longer term goals. Moreover, based on our contextual inquiry observations, we feel that a visualization of a few days in the future is usually sufficient to supply users enough information to support their tasks for today. Any detailed future planning should be done on the user's original calendar application, a more specialized system. The sidebar is meant to allow easy access to information that will help the user decide what task to do now. Specializing it to be a full calendar application would adversely affect the rest of the sidebar, an unacceptable alternative. By allowing the user to double-click through the pane to reach their default calendar, we hope to keep the connection between sidebar and the rest of the user's work flow tightly coupled without sacrificing *Stardust*'s focus on "right now."

Block and agenda views

The schedule pane consists of two mutually exclusive views, block and agenda. The block view shows the user's engagements as blocks of time with a size proportional to their duration, in similar fashion to the Google calendar day view. However, in contrast to a schedule like Google calendar, the top of our schedule pane schedule anchors itself a half an hour behind the current time. For instance, if it is currently 10:30 am, the top of the block view would represent 10:00 am. From this anchor point, the user can see the day's meetings and events with proportional gaps of space where no events are placed. This

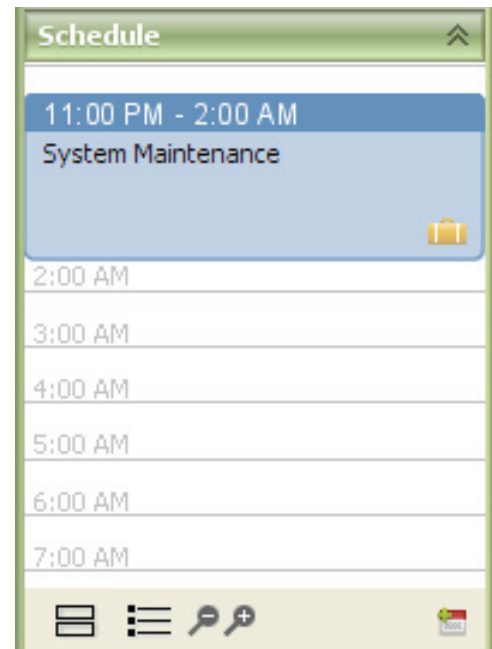
reflects a contextual inquiry observation that many users planned their to-do orders around the fixed items in their schedules. The block view has no scroll bar, since a scroll bar would not permit “now” to remain at the top of the sidebar at all times. Instead, the user can zoom in and out to see more or less time in whatever space is available while still seeing the titles for each event. Since the *Stardust* sidebar is designed to show information necessary right now at a glance, the block view allows the user to visualize the timing from now into the near future without having to manually adjust anything on the sidebar. Only when he or she wishes to zoom out and see more time does the user have to interact with the schedule pane.

Block view is very useful for planning a few hours ahead, but it also fills a large amount of space with empty time blocks. As the user zooms out, these blocks become increasingly wasteful, preventing him or her from easily seeing pertinent information. The agenda view shows a list of scheduled items for the next five days in a textual form similar to the Google calendar’s agenda view. By eliminating blank space, we allow the user to perform less fine-grained planning for several days in the future. Both our contextual inquiry and concept validation indicated a diverse set of user preferences when it comes to organizing schedules, so we allow users to easily switch between these two modes to obtain the benefits that they find most appealing, be it rich visualization or the ability to plan further into the future [V.1.b, V.2.b]. By scaling smoothly from great detail about one day to a summary of several days in the future while still keeping “right now” in a fixed place, the schedule pane is able to support planning while remaining flexible.

Given the time constraints under which our prototype was developed, only the block view has been implemented.

Anchoring

As described above, the block view enforces the concept of anchoring the top of the pane to one half hour before the current time. While settling on the idea of anchoring we came up with several competing options. One was to anchor the view at the start and the end of the



The block view of schedule pane

Schedule			
Mon 6/18	2pm	Conference call with Lindsay	📅
	4:30pm	Meeting with Jim	
Tue 6/19	1 pm	Lunch meeting	📅
	4 pm	Seminar	
	6 pm	Presentation	
Wed 6/20	10 am	Dentist's appointment	📅
	3 pm	Client visit	
	4 pm	Job interview	
Thurs 6/21	2pm	Mail Lisa company letter	📅
	4:30pm	Meeting with Jack	
Fri 6/22	9 am	Conference call with Ian	📅
	3 pm	Team meeting	
	4 pm	Committee meeting	
			<div> <div>📅</div> <div>📅</div> <div>🔍</div> <div>🔍</div> </div> <div>Add ▼</div>

(Wireframe) Agenda view on the schedule pane

day, allowing users to see their whole day at a glance, but in this view seeing event details and accommodating non-traditional hours becomes difficult in limited space. Another option is not to anchor the calendar at all and simply show several hours of the day along with a scroll bar. In this way, users have the flexibility of viewing any time of the day; however, they lose the benefit of the system adjusting to show the current time so that they can effortlessly plan the near-term future. Given the focus of the *Stardust* sidebar on the here and now, it seemed inappropriate to allow the user to move away from the current time. To this end, we considered anchoring only the top of the block view to the current time, but in concept validation, we realized that users may be running behind schedule and may want to see items in the past [V.2.b]. To solve this problem we adapted this option so that the top of the pane always shows half an hour behind the current time. In the end, we chose the one end anchoring scheme to ensure that users would be able to see how much time is left before their next scheduled item while still allowing them to zoom to a desired level of detail. Furthermore, this type of anchoring correlates best with one of the main initiatives of the sidebar: managing user tasks so they can work efficiently and accomplish more by helping them more easily decide what to work on next.

Zooming

Our navigation paradigm, zooming, is featured in both block view and agenda view. There are two buttons at the bottom of the schedule pane, a magnifying glass with a minus symbol and another with a plus symbol. Using the minus magnifying glass the user can increase the amount of information shown because the schedule zooms out to show a larger time range. Conversely, the plus magnifying glass zooms in to show a smaller range of time with more detail. Further, when the user zooms out beyond the point where block view is useful, block view becomes schedule view and vice versa upon zooming in. We settled on zooming to avoid using a scrollbar, which could move the current time out of the pane's preview, while still allowing the user to adjust his or her preferences for detail versus amount of time displayed. To counteract the problem of inflexibility that comes from an anchored schedule, we included zooming to be able to see later in the day in block view. However, in user tests with paper prototypes, we found that users tried using the zoom buttons to see the next day in their schedule. Therefore, we decided to put zooming in both calendar views for consistency's sake and have the block view switch to agenda view when the user tried to zoom out so much that the lack of detail would no longer be useful. In this way, it is always clear how to get to any level of detail in the the schedule pane within its time frame.

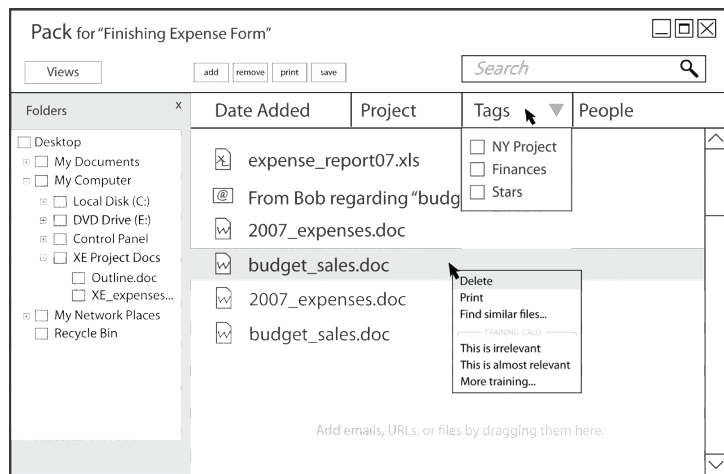
Editing event details

Much like a regular calendar, every event field in the schedule is intended to be editable by the user. We chose to make the fields in the schedule editable not only to be consistent with the task pane, but also to make it easier for users to deal with scheduled items directly from the sidebar. Since it takes no more space to make fields editable than the space it takes to display them, this design decision seemed obvious, although there was not sufficient time to implement it in our prototype. Ideally, any changes made in the sidebar would also be reflected in the original calendar that exists in the user's calendar application since it is still intended to be user's main long-term scheduler. Finally, as in the rest of *Stardust*, editable fields also teach the CALO AI by letting it know that something was wrong and providing the correct information while simultaneously providing direct benefit to the user.

f. Packs

Packs is a concept extended from SRI's PrepPAK, but was not implemented in this prototype. One major strength of the CALO AI is its ability to detect patterns and relationships in order to understand user's work. With this knowledge, the system can figure out what resources users might need while working. Our background research in information management and concept validations helped us define how such an AI system could best help the user [V.1.a, V.2.b]. The

"pack" feature in CALO *Stardust* automatically generates and dynamically updates a set of resources associated with a particular meeting, task or event for the user.



(Wireframe) The pack view that users can access by clicking on the pack icon on *Stardust*

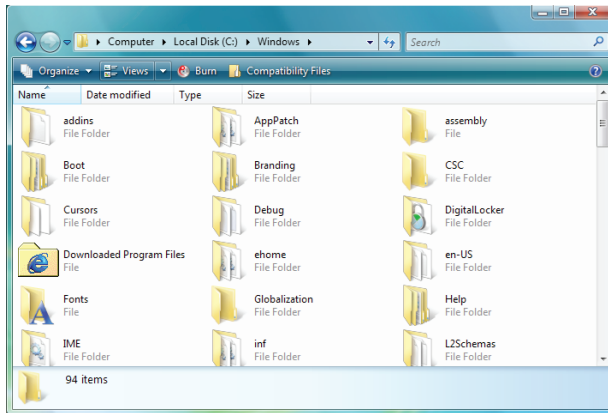
Pack view

A pack exists to collect and display all the resources that the AI considers relevant for a task, meeting, or event. For example, if the user requests a pack for a given meeting, the pack window would show files such as PowerPoint presentations, Word documents, links to websites, or other notes that user would need for that meeting in a pack window. The concept of pack is loosely based on CALO's "PrepPAK," which allows users to gather resources needed for meetings specifically. Backed by similar requirements from the AI, we expanded the concept to apply to a more general set of events. Upon testing our extended idea in a concept validation, we found it to be a very popular idea across all target users [V.2.b Fig 20]. Further, we found that in addition to meetings, users have many other tasks with associated artifacts and resources. Collecting these resources automatically resonated strongly with users.

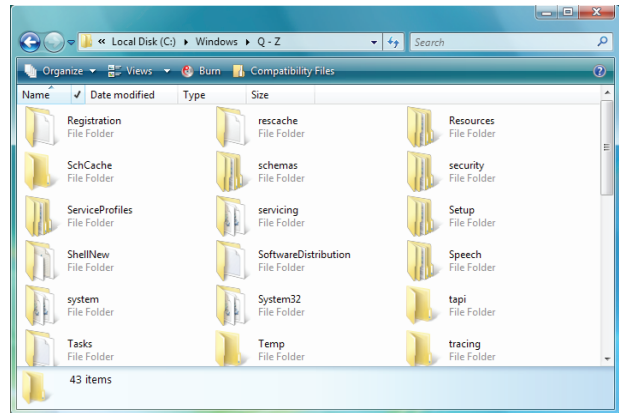
Consistency with Windows Explorer

Our team consciously intended the pack window's interaction and visual look and feel to be similar to a Windows Explorer folder window. The interface looks very similar to a Windows folder and the main mode of interaction involves selecting a file with a click of a mouse or selecting multiple files by pressing the "Ctrl" key while clicking on files.

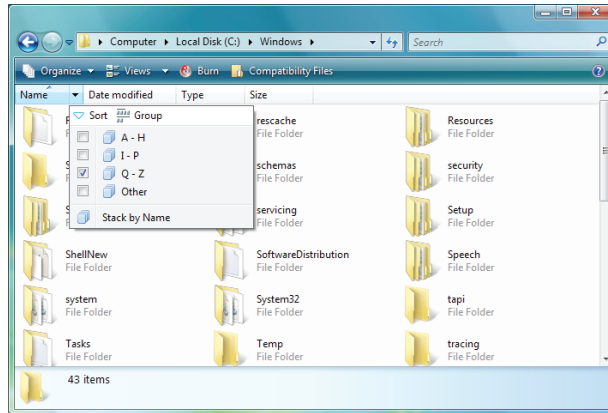
Our decision to design the pack window to look like a Windows folder stems from two major reasons. First, it leverages current successes in CALO Express, which uses a similar interaction for its PrepPAK feature. Second, much of the interaction with the pack includes filtering, sorting, and selecting files, use cases already supported in a recognizable way by Windows. By using known interactions, we can reduce the learning curve of the interface while still allowing the user to organize robustly.



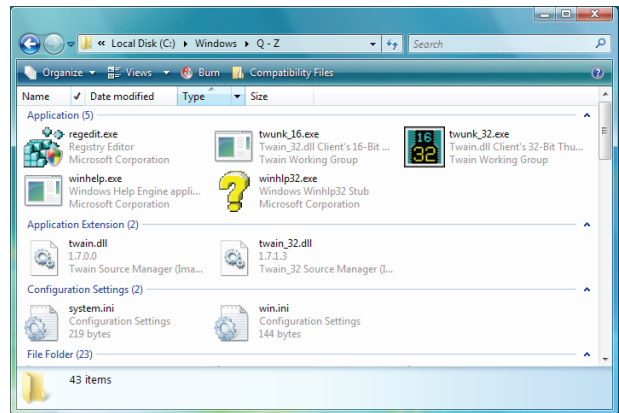
A Windows Vista Explorer folder



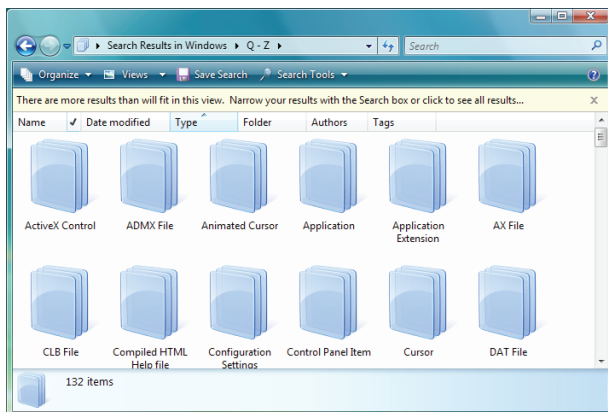
(Vista) A folder with an active filter



(Vista) Filtering by name



(Vista) Grouping files by type



(Vista) Stacking files by type

Users can sort items in the pack view on a number of different metrics by clicking on the columns at the top of the window. The user is also able to search for items using a Windows-style search field. We felt that it was important for users to be able to control the information in the Pack easily, because in our contextual inquiries we learned that some user tasks include handling and organizing a large number of files [V.1.b]. Therefore, we wanted strong filtering and sorting abilities in the pack to assist the volume of artifacts that they manage. Further, given that our pack view will be populated by an AI algorithm, we needed contingencies for users to sort out and remove the inevitable incorrectly added items. In the course of designing our pack view, we found that Windows Vista has powerful and intuitive filtering and sorting methods, so we modeled our interaction to be consistent with the Vista operating system. Our think aloud user tests with paper prototypes also confirmed that users were relatively comfortable with using the Vista-style filtering and sorting interaction [V.3.a].

Automatically updating items

Overtime, the pack view automatically updates to change the items that the AI thinks have associations with the selected task or scheduled event. This is powerful benefit provided by the CALO AI, but it presents some problems from users' standpoint. The items in a pack window for a given event can potentially be different any time users request a pack. As such, our team needed to design a way to allow the system to maintain dynamic flexibility while still making it clear how to save truly pertinent information. The first step is to allow users to modify the pack to actually meet their needs. If there are items that do not belong, users may remove them from the window and the AI will learn from user's actions. Likewise, users can manually add files into the pack and the system will learn the association. In both cases, users can use normal Windows file metaphors like delete and drag to accomplish their desired organization, supplemented by right-click training options should he or she desire to train the AI explicitly. In summary, the pack view usually uses AI to automatically change items in the view so that users do not have to update it themselves, but we also consistently present the pack view as a virtual directory that users can always edit on their own.

Representing items as shortcuts

The file system metaphor is useful for motivating our intended interaction with the pack view; however, our concept validation brought up one major problem—users are not comfortable working with real files unless they are absolutely sure where they exist on disk [V.2.b Fig 20]. Losing any file is a catastrophic failure. This perception was proved by later interviews: target users expressed their hesitation to move, remove, or add items to packs because they thought that they were actually deleting or moving a real file on disk. To assuage this problem, we decided to leverage another concept from Windows, the shortcut. Shortcuts exist to allow users to manipulate a pointer to a file—if it is moved or deleted, users know that the real file is still in its original condition. This is exactly the guarantee that we want for items in the pack view; as such, all the items in the pack now have a shortcut badge over their file icon to indicate that the items in the pack are actually shortcuts to the actual files in the directory. This makes it clear that when interacting with the pack users are managing a combination of resources, but not working directly with actual files.

Their actions will affect the pack and consequently the CALO AI, but they will never affect files on disk.

Saving

As described above, when users create a pack, the files are not yet in an actual directory, but represented in shortcut form in the pack window. The content of the pack can change dynamically, adding newly pertinent files, deleting newly irrelevant files, and resorting as the AI's understanding of the association changes. Further, users are able to customize the pack by adding or deleting files in the pack view window. Our concept validation indicates that such dynamism could be confusing to users who expect file explorers to change infrequently, and only in direct response to a user command [V.2.b Fig 20]. Therefore, we introduced the concept of saving a pack to ensure that a particularly useful pack's content will stay fixed. When users save the pack, they are deciding to keep a fixed copy of this specific combination of associated items saved into an actual folder. Thus, the pack is transformed from a virtual directory into a real folder on Windows, disassociated from the CALO AI for purposes of updates. The concept of using save to create an actual folder is similar to the "smart folder" concept used by the Mac application Spotlight, but at first we were worried that Windows users would be confused with by a menu item labeled "Save" in a window that already resembles a folder. However, we found in paper prototypes that users did not have major problems using or understanding the save feature [V.3.1 Fig. 31].

g. CALO suggestions

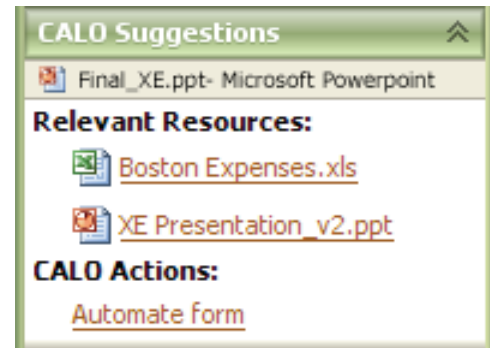
The CALO AI is currently able to deduce what users are working on by analyzing the windows that are in focus and presenting suggestions based on that knowledge. Its sophisticated AI can determine what users are working on and leverage its database of knowledge to help the users. In *Stardust*, we created the suggestion pane because, although there are many ways for users to interact with the system, there are very few ways for CALO to initiate action with users. The suggestion pane provides a way for CALO to present its suggestions and use its knowledge to help users. Placing the suggestion pane on the sidebar (as opposed to integrating it into individual windows) is also convenient for users because it makes it easier for users to learn that the same location always presents the AI's viewpoint. Moreover, it is far easier from an implementation standpoint than attempting to integrate CALO's interface into numerous third-party applications. Upon further examination, we realized that in the mini-bar, the items in the CALO suggestion pane are not visible, and users would not be aware of the actions CALO could perform. In this case, the CALO suggestion icon changes its color to indicate that there are new CALO actions available for users. Due to the lack of any real AI backing our prototype, the CALO suggestions pane is currently implemented only as a static view with no interaction.

Relevant resources and CALO actions

The CALO suggestions pane shows resources and CALO actions that are related to the window users have focused. For instance, if users are reading an email, then the CALO suggestion pane will show files, people, and websites that pertain to that email. When there

are actions that the system can perform, for example task automation, the actions will also be listed in this pane. According to our concept validation, relevant resources similar to those shown in the pack window are perceived as quite useful [V.2.b Fig 20]. Therefore, CALO suggestions also extends this concept by allowing a few highly-relevant resources to be easily accessible without creating a pack. CALO actions also exist in this pane because it provides the *Stardust* sidebar with its only way, outside the notification center, to directly express that it has learned from user's actions and to communicate to users directly.

Since demonstration of learning is another highly important aspect of the CALO system in general, the combination of visible resources and CALO actions serves the purposes both of users and of the development team.



The CALO suggestion pane

One final important aspect of the suggestion pane is that it displays a label indicating to what window its suggestions refer. For instance, if CALO presents files related to the PowerPoint presentation users are currently working on, then the CALO suggestion pane will have the name of the PowerPoint file at the top. We found in user tests that people were confused about what the files and actions in the suggestion pane were relevant to [V.3.a Fig 31]. Based on research into a similar interface problem by our client at SRI, we decided that simply indicating the window the AI was referring to would solve the problem most directly. Now users get relevant information and might be able to infer why the AI made the decision, better helping to demonstrate learning and improving the overall *Stardust* user experience.

h. Task automation

Currently, task automation is unimplemented in our prototype because it requires a serious AI backend to support it. Though only visible on the sidebar as suggested actions in the CALO suggestions pane, automation is an integrated feature of *Stardust* that plays a vital role in managing the overwhelming tasks our target users face. From our user studies, we found that users, especially assistants, could benefit greatly from the system assisting them in completing repetitive tasks, if not taking them over completely. Our concept validation, however, revealed the conditions limiting when users would accept and use task automation: first, we found that users wanted to preserve their clear control over the system decisions, and to correct mistakes easily [V.2.b Fig 23]. Secondly, they wanted the ability to have *Stardust* partially automate a task and then hand it off to them to handle any details of the task that change from instance to instance.

The CALO AI is already able to automate repetitive sequences of tasks via sophisticated learning algorithms. There are two ways for the system to learn to automate tasks: either by observing user actions over time, or from the explicit scripting from the users. There are also

two ways for users to access task automation, from the CALO suggestions pane and from the icon in the application access pane. The second option is suitable in cases users want to initiate task automation instead of receiving suggestions related to open windows. The CALO automation icon also provides a natural path to view the list of automated tasks and make adjustments.

i. Learning log

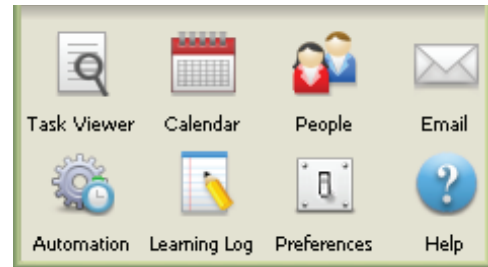
CALO's use of AI to observe and adjust users' tasks is a novel concept which prompted many users to want to track the progress of system learning. Although the notification center displays the system's actions, there should be a more permanent record for users to refer back to. *Stardust's* learning log serves this purpose as a repository for users and system actions, as well as a place where users can fix any mistakes made by the AI. Tracking system learning in turn requires tracking which user action led the system to learn particular associations. Thus, the CALO learning log was created to document both user and system actions so that users can check up on the AI's activities if they so desire. These documented changes are maintained at a finer granularity than the CALO actions that warrant a notification to the users.

By keeping users and system activities together, we are able to leverage context to explain system decisions. In our concept validation, users expressed the desire for an explicit place to train the system; however, they did not seem comfortable with AI jargon such as confidence level as an explanation of system reasoning [V.2.b Fig 23]. With that in mind, we thought it was logical to integrate documentation and training in case users did not understand why the system did something. Therefore, the learning log also serves as a place for users to train the system by correcting any incorrect associations the AI has made. By including contextual information, users may be able to infer the system's decision-making process more easily than they might be able to interpret a human-readable explanation of CALO's confidence level. While the user's inference may not be technically correct, the comfort gained by perceiving that he or she understands the system's reasoning is paramount. Additionally, in keeping with *Stardust's* ubiquitous editability paradigm, users can agree or disagree with CALO's conclusions and responses from the learning log to explicitly teach the AI. Manually training CALO *Stardust* allows it to learn much more quickly than simply letting the AI observe the user's actions over time. Though currently unimplemented, the learning log would be accessible through the icon well at the bottom of the sidebar, so that the comfort of seeing CALO's reasoning is always a click away.

j. Icon well for application access

The *Stardust* sidebar has a number of hooks into the expanse of the CALO system, and serves as a launch pad for interacting with the agent. However, there are use cases in which the user may want to access CALO applications directly. To this end, the bottom of the sidebar provides eight icons to bring the users to their task viewer, calendar, email

application, contacts list, CALO automation viewer, learning log, preferences and help. Note that most of these applications are internal to CALO, but a few (calendar and email) are the users' native applications. We created this pane so users could have easy access to frequently used applications, both inside and outside of CALO, with the hope that such ease of use would encourage users to attend to the sidebar more frequently and discover its



The icon well

other benefits. In our contextual inquiries, we found that overburdened knowledge workers frequently use their email client and calendar, so we kept this in mind while designing *Stardust* and included access to these high frequency applications in the sidebar alongside other applications that we want to make readily available to the users [V.1.b]. The CALO sidebar distills information to locations that help users get their work done in the moment. However, less common, long-range planning tasks are best completed outside of the sidebar. The icon well makes the transition between the sidebar and the rest of the users' world as simple as possible.

Several of the icons in the icon well allow access to applications that require no explanation (i.e. the user's native calendar, contact list, and email client as augmented in the background by the CALO AI), and the task viewer discussed in more detail above with the task pane. For a further discussion of the CALO automation view and the CALO learning log, please refer to their detailed descriptions above. The two icons on the bottom left of the pane, preferences and help, are less novel, but equally important. As a general HCI principle, it is well known that persistent access to help is absolutely essential to a usable system. In an AI system like CALO, the explanations available in help are all the more important.

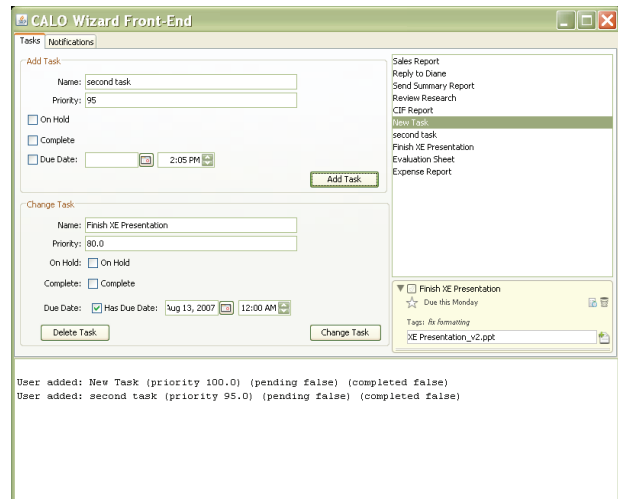
This is also the case for preferences. If our team learned one thing about executive work flows in our contextual inquiries and concept validation, it is that individual organization preferences vary widely [V.1.b, V.2.b]. CALO has an advantage over other systems in that it can learn and adapt itself to match the user work flow over time; nevertheless, for some essential settings like notification intrusiveness and autonomy, it is preferable to allow the user to set up preferences up front and at any time during their use of the system. The first few days with CALO will likely constitute a critical period for continued use—if users can specify preferences on set up and easily adjust them, it becomes far more likely that they will adopt the system. Our research indicates that most users would be willing to try out a system like CALO *Stardust* but would only continue if their initial user experience was clearly more beneficial than it was annoying [V.2.a Fig.18]. Providing users with upfront and accessible configurations is a big step in this direction. Along with obvious demonstration of learning, our team is confident that CALO *Stardust* will provide enough utility at a low enough cost to create an initial user experience that will keep CALO running and learning on target user's desktops.

k. Wizard-of-Oz Implementation

We designed CALO Stardust to employ SRI's sophisticated artificial intelligence engine to carry out tasks such as prioritization, automation and learning, yet integration with SRI's AI engine would require too many technicalities outside of the scope of our project. However, in order to effectively user test Stardust, AI actions need to be taken and seen on the Stardust interface. We decided to use a method called Wizard-of-Oz to simulate the actions the AI would have taken during our user tests to get the most accurate feedback on our design.

In a Wizard-of-Oz user test, the sidebar prototype is run on one computer for the user to interact with. Meanwhile, an experimenter uses a second computer to connect to and control the user's prototype to simulate the actions that the AI would have taken. In our user tests, the experimenter either sat behind the user or looked to a projection of the user's computer in order to see what the user was manipulating. When the user test called for an AI action, the experimenter would perform them from the second computer to change the user's interface on the first computer.

In order to perform Wizard-of-Oz, our programmers designed an engine that allowed us to mimic the actions the AI would need to take while users were interacting with our prototype. Since we only tested the system automation on notifications and tasks, the engine had two sections, one to control tasks and the other to control notifications. In the tasks section, the engine gave us access to add new tasks to the user's sidebar, set the tasks priority level and due dates, as well as edit existing tasks. The priority numbers assigned are relative to the priority numbers already in the task pane. The task will then be added or reordered in the correct place based on the priority number. The engine also allowed the wizard to delete tasks and move them to different subpanes. In the notifications section of the engine, the wizard is able to add and change notifications while also setting all the properties in each notification category.



The tasks section of the Wizard-of-Oz interface



The notifications section of the Wizard-of-Oz interface

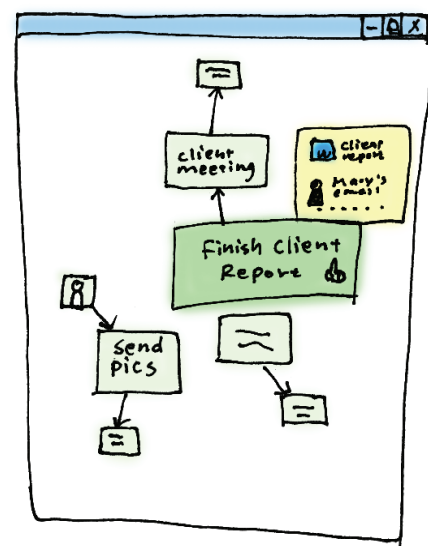
With the Wizard-of-Oz method, we were able to test many of our concepts and usability issues such as automatic task movement during prioritization, saliency of notifications, and general interaction between users and autonomous agents [V.3.b]. The method itself was sometimes difficult to use due to technical difficulties such as networking between two computers. The experimenter also had to constantly be aware of the user's movements and sometimes improvise to match the testing goals and specified interaction with the user's actions. For instance, if the user did not read an email that assigns the user a task, then the experimenter should not add the task because the AI would not add tasks until the user reads the email. Overall, Wizard-of-Oz allowed us to user test early in the implementation stage and without having to integrate Stardust with the artificial intelligence back-end.

IV. Future steps

Given our team's limited time and resources, there are components of CALO *Stardust* we envisioned but did not fully flesh out with details nor implement in our prototype. Most of these are visions that came up during our ideation that were beyond the scope of our project.

1. Task relationship view

During our contextual inquiries, our team noticed users need to visualize the relationships between their tasks [V.1.b]. These tasks depend on several dimensions such as resources, people, and simple ordering considerations and each have different priorities. The task relationship view is intended to be a graphical visualization of users' tasks and the connections and relationships between them, laying the tasks out across their dimensions of connection. Imagine a tag cloud over user tasks. The closer the tasks are, the stronger their associations and bigger the tasks are more urgent they are. The pending tasks are represented with connecting arrows to indicate tasks' dependencies among each other. We envision this view as an on-demand window accessible from the task viewer to facilitate the somewhat infrequent use case of planning out tasks in over a long period of time.



(Initial idea) A task relationship cloud

2. Automated skill transfer

CALO *Stardust* already adapts to individual user behavior, but our contextual inquiries identified further opportunities for the system to learn industry or company specific details and standards [A 1.b]. Such wide-ranging skills could potentially be transferable to other employees within a similar domain rather than having every employee spend effort training the same principles. For example, it would be very useful to transfer a skill like arranging travel expenses in the company that a CALO has learned through careful training by one user. Further, a robust skill transfer system reflects currently observed work practices; it is common for an executive to seek help from other people's assistants in the case that the assistant has some specialized knowledge. If CALO could reflect these extant practices, the AI may be able to leverage its training from the feedback of a number of individuals, increasing learning speed and accuracy, and thus improving the CALO user experience in a very general way.

3. Collaboration: assistant and executive

Collaboration is a large opportunity area that CALO has yet to fully tap into. One possible area is supporting the relationship between executives and their human assistants. Although CALO mimics many of the actions that an assistant would take, our contextual inquiry results indicate users would find it preferable if CALO instead focused on supporting the relationship between the executives and assistants rather than replacing the need for assistants all together. Through our research, we've found that assistants usually have access to the executive's calendar and email in order to be able to better serve and organize them. By giving assistants access to an executive's CALO, an assistant would be able parse the information constantly coming to the executive and share these important tasks and events directly through the sidebar. By better supporting repetitive components of assistants' work flows, CALO can free them up to concentrate on the more important, person-coordination aspects of their job. In this way, agent and assistant can work together to raise productivity for everyone for which they are responsible.

4. CALO mobile

Our final vision, CALO mobile, is a hardware concept that addresses the portability of information CALO holds as well as the screen real estate that the sidebar takes up. CALO mobile would be a PDA type device that could synchronize with user's monitors. While docked, the PDA would act as a second screen that users could seamlessly interact with from the main monitor. The second screen would display the sidebar for users to add tasks and resources to. When users are away from the desk, CALO mobile is still active to remind them of time sensitive events along with constant access to important information. Coupled with other capabilities like CALO's Meeting Assistant which automatically transcribes meetings and creates action items, CALO mobile could be a impressive way to integrate

V. Appendix

1. Research

a. Literature review

CALO is a highly complex system, the design of which draws upon many well-researched topics in computer science, human-computer interaction, and other fields. Our research goal was to familiarize ourselves with these topics and determine what impact each research area has had on the development of CALO up to this point, and how this research is likely to continue to affect it in the future.

Based on our exploration of the CALO publications and those of related projects, the four themes that appear to have most directly impacted CALO's development can be grouped into collaboration, information management, agents, and multimodal interfaces. The applicability of research into agents is certainly not in doubt given the nature of CALO, and that collaboration and information management should be a major factor is evidenced by the fact that many of CALO's major subsystems (such as the meeting assistant, PTIME, and Towel) are all information management tasks geared at least in part toward facilitating collaboration between CALO users. Multimodal interfaces are not particularly integrated into CALO at this point in time, but they are associated with context-aware computing and software agents in the literature for some time, and there seems to be considerable interest on the part of the developers in making multimodality an increasingly important part of CALO's user interaction. For this reason, we considered them to warrant further study as well.

Collaboration

In our review of literature pertaining to CALO, we first focused on the ways people collaborate with each other and the ways that software (especially agents) can support this interaction. As early as 1990, Grosz and Sidner [21] theorized that successful collaboration stems from mutual understanding about the goals, action, capabilities, intentions and commitment of the participants who collectively form a shared plan. However, Barthelmess et al. [4] observed that current collaborative technology is disruptive and does not support natural human to human communication. Rather than a series of user-command and system-display turns, they suggest a system with several unobtrusive sensors that recognize and process interaction in the background while creating appropriate artifacts. Multiple modalities are explored to construct a system that proactively identifies user's intentions.

One large area of interest in collaboration with human users is the realm of intelligent interruption management. From a theoretical angle, Iqbal and Bailey [24] examined the feasibility of building statistical models that can detect and differentiate three granularities of perceptually meaningful breakpoints during task execution, without having to recognize the underlying tasks for determining the breakpoints for optimum interruption. Fogarty et al. [19] used a variety of sensors to improve an agent's ability to interrupt human users

at more appropriate times. In an experiment with users performing a programming task, the sensor-based system was able to determine interruptability correctly 72% of the time, and the research discussed what sort of sensors are most useful for inferring context in programming tasks. Such a system may be useful in CALO, but more research would be necessary to determine appropriate sensors for context-awareness in non-programming situations.

Avrahami and Hudson [2] found that IM conversations offer several benefits for users, including the ability to selectively attend to or ignore messages, but are typically highly interruptive and do not allow users to easily prioritize important questions and information above less important messages. They successfully implemented an IM plug-in to determine whether incoming messages were likely to be important, and choose whether to interrupt the user depending on this determination. Adamczyk and Bailey [1] devised an interruption management system for monitoring and specifying user tasks using physiological measures of workload and task modeling techniques in order to systematically and automatically identify opportune moments in a user's task sequence to mitigate the negative consequences of interruptions by notifications.

Another large area of collaboration research is in ways to support cooperation via email, a heavily overburdened system. Dabbish, Kraut, Fussell, and Kiesler [15] proposed a model of email use to predict whether or not a given email will be replied to, with some modeled factors including using inbox visibility for reminders, keeping information requests and responses in the inbox, and responding to but not filing meeting requests. Using action requests, status updates, reminders, information requests and responses, scheduling requests and responses, and social content as proposed email types, the authors were able to use a regression model to match email importance to likelihood of response. They found that content, job complexity, and sender characteristics are good indicators of response, and that the identity of the responder plays a larger than expected role in predicting filing due to the tendency of people to either sort or search their email but not both. Again leveraging email's multiple uses, Shen, Li, Dietterich, and Herlocker [43] described TaskPredictor, an application that uses Naive Bayes classification and confidence thresholds to try to guess what task a user is currently performing based on what system resources are currently being used. Tests on a corpus of email showed a promising 80% classification accuracy that could possibly be improved through more computationally expensive methods such as Hidden Markov Models.

A final area of collaboration support is in the realm of agents to help schedule and document human meetings. Faulring and Myers [18] presented Rhaical, an intelligent calendaring system that proposes natural language support and novel visualizations to help users when scheduling meetings for multiple parties. The agent might contact users to verify assumptions, get confirmation, and allow the user to understand and control its behavior through natural language processing and manipulation of a calendar visualization. For actual meeting documentation, Ehlen, Niekrasz, and Purver [17] described the CALO's Meeting Assistant. This assistant analyzes multi-party speech and handwritten input, identifying the meeting's topics and action items, and displaying a high-level summary

report (along with the user's own manual notes) in a browser, so as to compel users to make manual corrections, and to suggest action item transfers to other agents such as the Towel to do manager. The combination of user feedback, integration of analyzed input and manual notes, and collaboration with other agents produces a highly personalized representation that parallels the user's perception of the salient aspects of the meeting.

Information management

Information management, including the organization and retrieval of information, is an increasingly complex problem as the amount of data users encounter continues to grow. Several tools have been developed to support users and their data such as email, to do lists, and calendars, but these applications are often overloaded and have weak boundaries. To address this problem, information management has been moving from manual tasks done by the user to more agent-based applications for automating processes. Several parameters have been studied, such as the hierarchy knowledge workers employ to organize their information that would better support their work flow in the contexts of email and file folders. Research also spans on other applications that are designed to convey information, send reminders, handle task management and regulate scheduling.

Boardman and Sasse [11] studied information management across tools, specifically files, email and Web bookmarks and long term issues relating to personal information management. They found that the nature of acquisition varied between tools from manually done in files and bookmarks to uncontrolled in email. File management strategies also varied from file on creation to file on completion of task or during a "spring cleaning." Similar patterns were found in email management where they found no filers who do not organize and instead search their email, frequent filers who file as emails come in and spring cleaners who file their email from out of their inbox at intervals.

Martin and Jose [37] reveal the other software that facilitates information management includes information retrieval system prototypes such as Fetch, which adopts the concept within an information-seeking environment specifically designed to provide users with the means to better describe a problem they don't understand. Along with Fetch, another piece of software, created by Bao et al. [3], FolderPredictor, works in the same problem space. FolderPredictor applies machine learning algorithms to the observation of users' opening and saving of files, analysis of document content, and the making of context-aware predictions to reduce the amount of time users spend locating their files.

Henderson [22] looked at the attributes knowledge workers use to structure their information into hierarchies. Genre, task, course, topic, time, and person were the most frequently used folder types. Some of the dimensions like person, source, topic, time and file type can be automatically supported by software whereas genre, course/task and security are unsupported by software automation and must be done manually. Lerman, Gazen, Minton, and Knoblock [33] used automated grammar generation, a automated technique to group data into hierarchies, to semantically mark up data-filled websites and tag them based on heuristics. This method labeled automobile sales data columns correctly 64% of the time, but inconsistent data formats and similarly formatted but semantically unrelated fields remain as hurdles to greater accuracy.

Other parts of information management include the use of email. An interesting example is the U.S. Government investigation into the Enron collapse which resulted in a large corpus of email messages analyzed by Klimt and Yang [30]. They found that while most users make use of folders to organize their email, it is also important to classify messages by thread and relationship to other messages, a difficult problem for a computer if human users fail to use 'reply' to maintain thread relationships. They also discovered trends in the data that indicate useful ways of classifying messages using the message body and from fields.

Besides organizing information, applications like email take on multiple roles of information management. Bellotti, Ducheneaut, Howard and Smith [5] recognized the transition of email as a task management tool supporting to dos, ongoing correspondence, delegation and receiving of work. To address the growing complexity of email, they created Taskmaster, an email system designed for task and project management. In their research they identified the following seven problems and designed Taskmaster accordingly. Taskmaster works by "keeping track many concurrent actions (the user's and the ones expected from others), making important things salient amongst less important items, managing activity over time (keeping track of threads of activity and discussions), managing deadlines and reminders which can be associated with other content, collating related items and associated files and links, application switching and window management, and getting a task oriented overview rather than a glance through scrolling or inspecting folders."

Some similar functionality already exists in CALO. Its front end, IRIS, is the interface that integrates all the different components of CALO's intelligent agent system for the user to operate. Cheyer, Park, and Giuli [12] summarized the concept of semantic desktops, intelligent knowledge management and systems for augmenting the performance of human teams and how IRIS was designed with components borrowed from existing semantic desktops and knowledge management software.

Conley and Carpenter [14] presented Towel, an intelligent to do list manager developed under CALO, is another tool somewhat similar to Taskmaster that handles task management with direct communication with the user. A style of digital communication between the user and Towel is vital to Towel's operation and training, and instant messaging's model of interruptions (opening a chat window and playing sounds), status information for different contacts, contact list (strikingly similar to the look of a to do list), and flexibility to carry out either rapid human-to-human dialogs or lax conversations (such as hiding the chat window until a more appropriate time) provides an ideal framework for the workings of a to do list. Also, the chat windows limit the number of operations the user can undertake, and they also make users directly manipulate an operation using commands, so to avoid dealing with anything outside of the context of the task.

As for action items that are verified and managed by Towel, they reach the user again in the form of notifications and reminders. In *Effective Interaction Strategies for Adaptive Reminding*, Weber and Pollack [40] discuss that a robust reminding system should consist of a motivating justification, attention to reminder granularity, user's preferred signal,

and machine learning techniques. There are two general approaches that these learning techniques have taken: one called reinforcement learning where the machine refines its reminding algorithm based on a cumulative reward system, and the other called supervised learning where the machine selects and presents certain data to the user for training.

Other applications that also share this management space include calendars. Modi et al. discuss [38] the CMRadar calendar management component is capable of making autonomous scheduling decisions, negotiating schedules with other users and agents, and prioritizing existing meetings to determine how to resolve scheduling conflicts. CALO's calendar component, PTIME, is an agent-based scheduler that learns, as Berry et al. [9] discussed in their paper A Personalized Calendar Assistant. Some features include the ability to work with the user to solve infeasible scheduling problems, automated preference learning and automatic inferences about when best to interrupt the user, backed by active, procedural, and especially reinforcement learning techniques. Later, Berry et al. [8] discusses the PTIME system is organized around the principle that people dislike giving up control over their schedules, whether to a software agent or otherwise. Since users often have widely differing preferences and practices in regards to time management, PTIME is designed to support and augment, rather than replace, the user's natural processes. Berry, Myers, Uribe, and Yorke-Smith [6] built this scheduling agent based on soft constraint-solving, allowing the system to autonomously create goals and reason about user commitments. They suggest that good constraint-based scheduling algorithms to handle scheduling already exist, but inherent uncertainty in user schedules requires even more robust responses to dynamic schedule requirements if a satisfactory system is to be fashioned.

Both of CALO's scheduling assistants, PTIME and Pisces, take the collaborative approach, called Mixed-Initiative, of balancing scheduling algorithms and human evaluation of schedule quality and nuances of domain constraints. PTIME is designed to learn and refine the user's preference model, whereas Pisces is more focused on providing solution to very large and complex problems. Berry et al. [7] expressed the hope that the scheduler's autonomy will grow with time, and indicated that reinforcement learning may be the best candidate to make this hope a reality.

Agents

With the expansion in amount of information people deal with on a daily basis and the advance of AI technology, computer agents are increasingly being incorporated in user interfaces. The notion of a cognitive agent that dynamically accommodate to user's work flow cannot be made possible without AI agent components.

Mike Papazoglou [39] summarized four different types of agents: application, general business activity, information brokering, and personal agents. Other types of agents generally fall under these main types. Application agents are application specific agents that are specialized to a single area of expertise and work cooperatively with other agents to solve a complex problem in the domain. A procurement agent is an example of an application agent. General business activity agents take care of typical commerce

transactions such as business purchasing, billing, parsing information on the Web, and finding trading partners. Information brokering agents (also referred to as matching agent) “maintain, update, and access distributed directory services,” as well as performing advanced navigation services. Brokering agents help service distributors publish their services and customers to look for these services. Personal agents work for specific users and their needs “to support the presentation, organization and management of user profile, requests, and information collections” distributed on the Web and the personal computers. Personal agents need to monitor and learn user habits and activities and may suggest better ways of performing these tasks. Examples of personal agents are intelligent tutoring systems and Web browsing assistants.

All computer agent designs must be designed with theoretical considerations and practical concerns to be successful. Through her study on interaction between users and cooperative AI agent that can initiate communication, monitor events and perform tasks, Maes [35] raised important issues related to topics such as agent personification, mental models, styles of training, privacy of users, and the responsibilities of agent’s actions and transactions.

Kaye and Karam [29] presented a design for distributed cooperating knowledge based assistants that emulate the behavior of human office assistants. These assistants cooperate with each other to complete tasks initiated by the user and interact with conventional office systems such as databases and message systems. The purpose of the agents is to relieve office workers from having to learn and use a large variety of systems or having to integrate tools to build high level applications.

Rich and Sidner [41] stated that autonomous agents should be governed by the same principles that underlie human collaboration and communication during shared tasks. Computer agents have varying degrees of autonomy determined by the granularity of the task and user’s needs. Usually, the user of the agent decides how much of a task to delegate to the agent. Alternatively, multiple-agent systems might identify different components of a task and delegate them to other agents inside the system. In a more interesting case, Maheswaran, Tambe, Varakantham, and Myers [36] discussed the concept of adjustable autonomy—the ability of an agent to decide when to cede control to a human user or to ask for confirmation. Schurr, Varakantham, Bowring, Tambe, and Grosz [42] examined the adaptation of Isaac Asimov’s laws of robotics to teams of autonomous or semi-autonomous agents. They found that, perhaps contrary to expectation, rigidly following human orders at all times leads to degradation in agent team performance and an increased, not decreased, likelihood of bringing harm to humans. This effect can be mitigated by communicating the agents’ misgivings with orders so that users can suggest alternatives. More research needs to be done to understand miscoordination costs among groups of human users and agents or in situations with uncertain knowledge states.

As for agent training, computer agents could possibly be trained in a similar manner as human assistants to human assistants. Agents can be trained explicitly, by observation and imitation, and by receiving positive and negative feedback from the user. The challenge is for agents to learn correct sets of information and provide enough feedback to the user so

that they could un-train incorrect assumption. Tambe et al. [44] discussed a set of semi-autonomous personal software agents termed “electric elves” placed in a work environment resulted in increased efficiency but also in several large social and work flow breakdowns when the user was unable to correct the agents’ faulty assumptions.

Kozierok and Maes [31] point out that both memory-based learning and reinforcement learning approaches would allow users to build up trust with the agent as it learns the user’s habits, making suggestions and predictions, coupled with explanations and confidence levels for the user to verify. Empirical testing indicates that the user-agent pair is more effective at the given task than a pair of human users.

Garera and Rudnický [20] discussed an agent designed to help users create weekly summary documents by making inferences from raw data as opposed to finished text. Despite this difficulty, a system trained on hand-classified data helped users complete the summary task in 22% less time over the course of the study. Unfortunately, automatic classification was less precise, leading the authors to suggest direct instruction, information synthesis, and active information acquisition as future supplements to improve the system. Tomasic, Zimmerman, and Simmons [45] aimed to create an agent that could help users find and fill in forms in complex corporate knowledge bases. Using natural language processing, and user-agent feedback loop system, the agent was able to retrieve the correct mini-form 80% of the time, an acceptable rate given the difficulty that humans have with this task.

To do lists prove to be a challenge to intelligent user interfaces, as presented by Gil and Chklovski [13] in terms of having to map users’ natural utterances to internal task representations, anticipate minor and preparatory tasks to accomplish users’ tasks, to determine the context of the tasks and know its own limits, and to know when task automation is desirable. The structure of BEAM includes the syntactic parsing of the user’s natural language in reference to several repositories of external and internal organization knowledge, and the collaboration with other agents (such as SPARK) within the CALO architecture in order to execute automated tasks.

The CMRadar agent presented an integrated component of Outlook and provided a novel interface for explaining its scheduling decisions to the user. It also established an interesting paradigm for multi-agent interaction—how agents communicate with each other entirely through emails. Modi and Veloso [38] also demonstrated multi-agent scheduling and rescheduling—how an agent takes into account the density of another user’s schedule to access the difficulty of scheduling a meeting with that person.

With regard to transfer of knowledge inside a single or between multiple agents, Marx, Rosenstein, Kaelbling, and Dietterich [16] discuss that knowledge transfer is profoundly complicated because the decision boundaries for different tasks exist in different feature spaces. Through an experiment where they observed the processing of two separate tasks, they found that while a machine can find the model of the first task to predict the parameters of the second, this only happens accurately if the tasks were generated from a

common source and existing in the same domain.

Multimodal interfaces

Multimodal interfaces—that is, the use of multiple modalities such as speech, gestures, written commands, etc.—have long been considered prime candidates for the interfaces of agents such as CALO. Several authors have explored the various strategies and ramifications of providing multimodal input, as well as specific opportunities that lend themselves particularly well to multimodal interfaces. Lunsford, Kaiser, Barthelmess, and Huang [34] described a set of “extrinsic costs” that are incurred when humans, who naturally interact multimodally, are constrained to unimodal computer interfaces. These include the need to over-specify or re-specify input to satisfy the computer, and the overhead of the interface misinterpreted user behavior. They also discussed ways that multimodal interfaces can reduce or eliminate these costs.

Huang and Oviatt [23] showed that multimodal input is often sequential rather than simultaneous, and that the choice of sequential or simultaneous input is very consistent within users. Some users were also observed to consistently choose unimodal methods of input even when multimodal input was available. Further, Krause, Siewiorek, Smailagic, and Farringdon [32] showed that physiological information such as stress level and movement patterns can be used to predict interruptability and determine the context of the user’s interaction with a wearable computer. The authors also made the point that non-intrusiveness and minimal active training are both essential features of a successful context-aware system.

Kaiser [27] explores new methods to supplement speech recognition by combining it with handwriting analysis rather than lip-reading and relying on mutual disambiguation techniques to acquire out-of-vocabulary words. In baseline test, detection rate of new words was 100% with greatly improved error rate and accuracy metrics; however, the test data set was too small to make any definitive conclusions. Kaiser et al. [28] created Charter, a system developed to support remote collaboration. Charter used multimodal sketch recognition, vision based body-tracking, and speech/writing recognition for minimal intervention on work practices. In this system, the inputs can be displayed to distributed members in other locations. Charter can learn new terms used by the group and build semantic interpretations based on interaction.

Biehl and Bailey [10] studied comparing how well three classes of interfaces, textual, map, and iconic, support application management during realistic, collaborative activities in a multiple-device environment (MDE) and found that users preferred and performed better with the iconic interface due to its more comprehensive visual and spatial representation.

In another paper, Kaiser [25] discusses the SHACER (Speech and HAndwriting reCognizER) software’s capabilities of learning new terms dynamically from single human-to-human interactions during multi-party meetings, applying knowledge of persist across related meetings, and determining the semantics of handwritten abbreviations. Lastly, Kaiser, Demirdjian, et al. [26] demonstrated the collaborative creation of Gantt scheduling

chart using multimodal interfaces including gesture recognition, handwriting recognition, natural speech processing and body tracking.

Research summary

From our research, we have identified several areas of key research interest relating to CALO and many insights from previous work in these areas. In particular, we see that current collaborative technology can be disruptive to the very collaboration it is meant to support, and that interruption management may play a key role in mitigating this effect. We also note that people tend to leverage existing technologies such as email and instant messaging and overload them to take on new responsibilities and tackle new tasks. Agents like CALO must take this into account and seek to leverage existing technologies like these itself.

We have also discussed several extant examples of cognitive agents, some of which have been deployed and observed in the field. Agents clearly fall into many different categories, all of which behave somewhat differently. The design and implementation of an agent must take into account social as well as technological factors, since agents often take an active role in their users' social environment. We also see several different approaches to human-agent interaction, with varying degrees of autonomy. We also observe differences in mental models and training styles, and varied approaches to the degree of personification expressed by the agent. We note further that multimodal interaction is of central importance to CALO and other cognitive assistants; research indicates that multimodality can solve or reduce the impact of many of the problems we have discussed, by reducing or eliminating many of the extrinsic costs of interacting with a computerized agent. Multimodality also has direct relevance to collaboration, since a multimodal interface can integrate much more tightly into a highly collaborative setting with minimal intrusion.

Much of this research was an interesting exploration of how collaboration is managed in a professional environment and what an agent could be capable of, but it was still unclear to us how this would apply to our target user group in the context of their work. We therefore embarked on a series of user studies to supplement this research, which is described in the next section.

b. Contextual design

User pool justification

To understand the needs of CALO's target user group, overburdened knowledge workers, we looked at two user types: assistants and executives. Executives fit the demographics of busy professionals who face the complexities of dealing with multiple projects and people at any given time. Assistants are a secondary user group identified because of their relationship with and importance to the primary target user group, executives. The assistant's job is also to focus on the work flow; therefore, they are better able to describe the mechanics of their work whereas executives focus on a high-level view and tend to disregard irrelevant details. Finally, in order to understand the use of CALO by people with either substantial training

or experience using the system, we obtained data from the CALO developers. Data from these individuals not only provided us valuable insights into how CALO is incorporated into users' actual work practices but also the perspectives from which different developers approached the problem domain.

After extensive focus setting sessions, we came up with two main areas of focus to direct our contextual inquiries: first, how do people collaborate on the job and what software supports this? And second, what mental model should CALO support to meet user needs?

Contextual inquiry overview

To collect data on our user groups, we conducted contextual inquiries to obtain insights and breakdowns about their work flow. Contextual inquiry is a method in which the researcher goes to the user's workplace to learn and understand his or her work in the context in which it lives. After a contextual inquiry is conducted, the entire research group meets to create models of the data collected. There are 5 models: flow, cultural, sequence, artifact and physical, which are created to reflect different, but important parts of the user's work. The flow model captures the responsibilities and work processes in the user's job. The cultural model records influences that come from groups or organizations that the user perceives onto themselves. The sequence model documents the steps and procedures the user takes to accomplish his or her tasks. Artifact models are representations of actual documents that the user uses in his or her work flow. Lastly, the physical model is a map of the user's physical workspace to capture where the user works and its effect on the user work flow. These models are created for each individual user and then consolidated by user type to gather insights about the user group rather than individuals and their details.

Our contextual inquiries

We conducted 14 contextual inquiries over a three month period with our three user groups: assistants, executives and CALO developers. Within the 14 contextual inquiries, there was some overlap between executives who were also developers.

Consolidated user models

After gathering a large amount of data at the granularity of a single CI user, we consolidated the models that we generated in order to visualize the data at the level of a user archetype. In this way, we are able to factor out the idiosyncrasies of individual users and design from more general trends that will support our user base as a whole. Since CALO must serve a number of very different users in different ways, we decided in this case that it would be most instructive to consolidate our models into three user archetypes—the developer, the administrative assistant, and the executive. These specific archetypes were motivated both by the groups of target users specified by SRI and our modeling process. Each archetype provides us with a varied set of insights and requirements for our design. They also served to motivate our final focus.

SRI developers

Our trip to SRI's main campus in Menlo Park, CA, provided us with both a number of insights into the ways in which end users might interact with specific parts of CALO and a

high-level overview of the ways in which the developers envision integration for CALO as a whole. However, as indicated on our developer flow model, their interaction with CALO was generally more limited and artificial than one would hope to see with an end user. Typically, a developer would focus on training and using the part of CALO that they were actively developing, more as a debugging procedure than as an actual user. As such, their interactions were more hypothetical than the sort of data typically observed in contextual inquiry.

While the data gathered was most instructive from a CALO-demonstration perspective, it was interesting to note that developers tended to struggle with the components of CALO that they were not actively developing. This indicates that in its current form, CALO requires too much low-level knowledge to operate, a problem that we hope to address through our observations of less technical users in the field.

Administrative assistants

During the consolidation process, we found that while the cultural, physical, sequence, and artifact models were consistent among all assistants, the work flow models differed to such a great extent as to imply the existence of two archetypes. The differences centered around whether the assistant was fully responsible for a small number of executives or was responsible in a more limited way for a larger number of lower-level employees. We named these archetypes the secretary and the coordinator, respectively. Once we made this distinction, we were able to draw a number of important insights from our completed models.

Constant interruptions. Our first interesting discovery is the observation that while assistants are constantly interrupted, elimination of their interruptions is not a viable goal for CALO. Instead, we see that these interruptions are an integral part of a work flow that is based around serving a large number of people for relatively short amounts of time (Fig. 1). Thus, instead of reducing these interruptions, we should focus on ways in which to support sequences that are resilient to interruption.

Waiting for others. Another work flow aspect that leads to a number of breakdowns is the necessity of waiting for external information. Commonly, this information comes from people, not databases, so the assistant is required to wait for the provider to actually get around to responding to their request (Fig. 2,3). The end result of this waiting is to fragment work sequences and cause the assistant to handle many tasks in parallel. This makes task prioritization difficult since it is not possible to simply follow one task through to completion.

Because of the difficulty of prioritizing tasks and a need for flexibility, the most common practice is to keep these tasks either on paper or simply in the mind, a set up that is prone to errors (Fig. 4). The problem is amplified for coordinators who have to deal with an even larger number of constituents who may be distributed across the office or further. Ideally, CALO will be able to serve as a repository for these sorts of short, pending tasks.

Extra responsibilities. The next insight, taking on responsibilities outside of one's job description, appeared with almost every assistant interviewed. It seems counter-intuitive to think of going outside of one's job description as an intrinsic quality of being an assistant, but the reasons behind such a phenomenon are equally as interesting. Over the course of working, assistants gain knowledge in some specific domains such as purchasing or making travel arrangements. This knowledge makes them a resource to employees for whom they are not directly responsible.

Interestingly, the executives for whom the assistants are responsible seem to encourage this behavior, "lending out" their assistants to perform tasks for their clients and office mates (Fig. 3). The cultural tendency of many assistants to be unable to turn down requests for help also exacerbates this propensity, which at times leads to feelings of being overwhelmed. CALO probably cannot directly support outside responsibilities, but there exists an interesting parallel between lending out one's assistant and skill transfer by the CALO agent.

Desire for perfect knowledge. The insights discussed so far all relate to aspects of the assistant's experience that greatly increase their work load and level of stress. As such, assistants are typically highly overwhelmed, and develop coping strategies to deal with this. By far the most prevalent is to seek "perfect knowledge" of the work of which they are a part. We observe that assistants try to know everything that is transpiring in their realm of influence, whether or not it is useful or relevant to them at that moment, due to their perception that they are the "last line of defense" for those who depend upon them (Fig. 3). They perceive that if they fail to take the appropriate actions in response to any external event, no one else will be able to correct their mistake before it has dire consequences. This perception also motivates the assistant to double-check everything they themselves do, to ensure that nothing has slipped through the cracks. CALO, acting as a repository for organizational knowledge, can both support this desire explicitly and reduce the cognitive load on the assistant.

Trust over time. In general, the executive and assistant relationship is one of increasing trust and responsibility over time (Fig. 3). Assistants tend not to be explicitly trained, firstly because there is insufficient time, and second because it is not always clear what the assistant should be trained to do. Instead, we typically see an assistant's functions expanding organically over time with increasing autonomy for them to manage items such as their executive's schedule and travel arrangements. This relationship definitely ties to the concept of adjustable autonomy in CALO, and warrants further exploration.

Frequent use of databases. One notable difference between secretaries and coordinators is the tendency of coordinators to interact with databases on a regular basis. Therefore, this interaction likely results because coordinators are responsible for supporting a larger number of people than secretaries, and databases facilitate handling many employees. Breakdowns arise because many coordinators are not particularly technical, and they treat these databases as "black boxes." Further, assistants tend to duplicate effort when asked to input transfer paper data into the database (Fig. 2). CALO's task learning component would likely be useful in reducing the burden on coordinators interacting with databases.

Consistent support. A final insight from our data on assistants is that while the executives they support are very different, assistants tend to support them in consistent ways. Some common activities are scheduling meetings, handling traveling arrangements, managing financial transactions, and providing reminders (Fig. 5). It may be most advantageous to design CALO to support assistant work flows because the applicability of such an approach

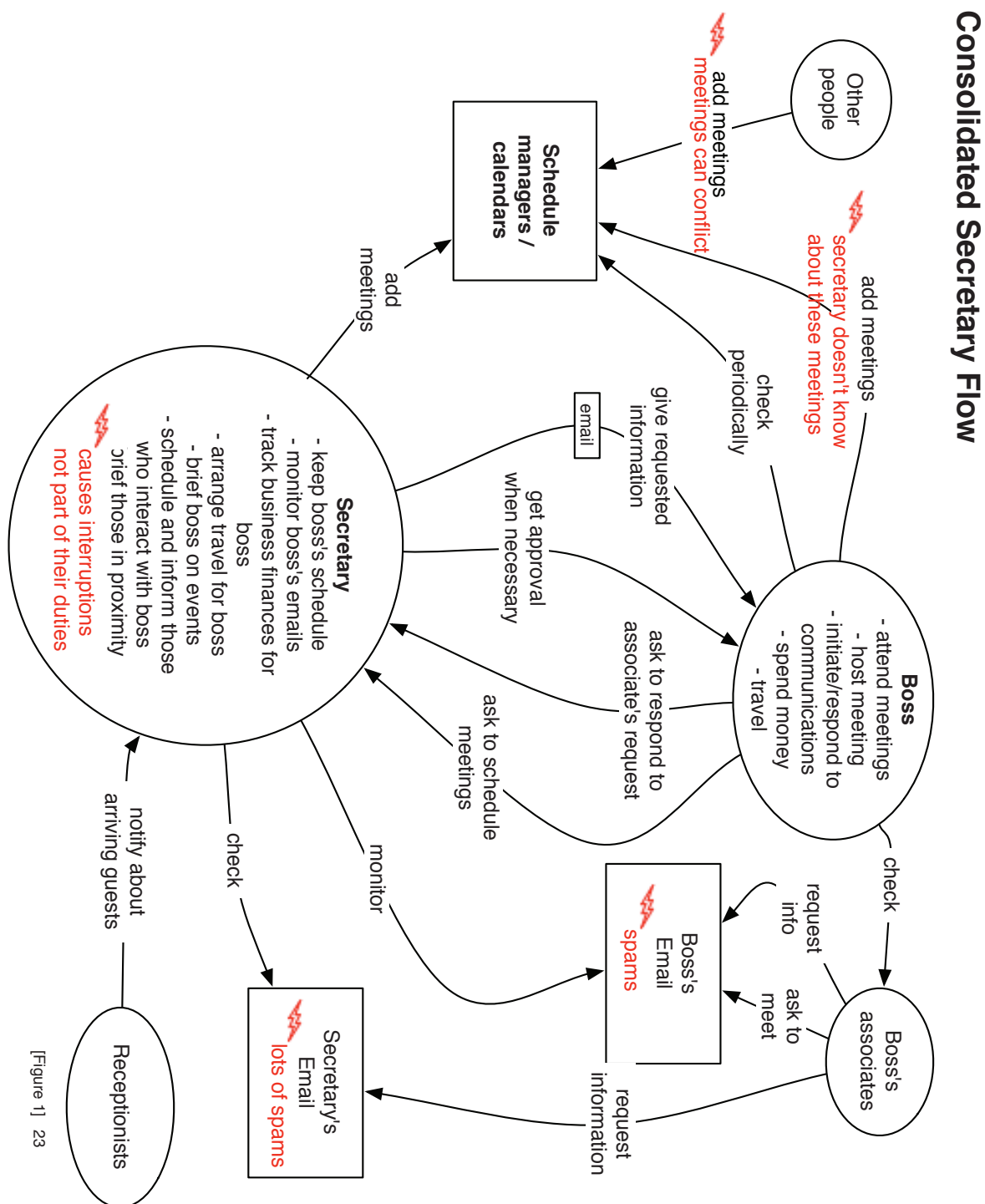


Figure 1

Consolidated Cultural Model – Assistant

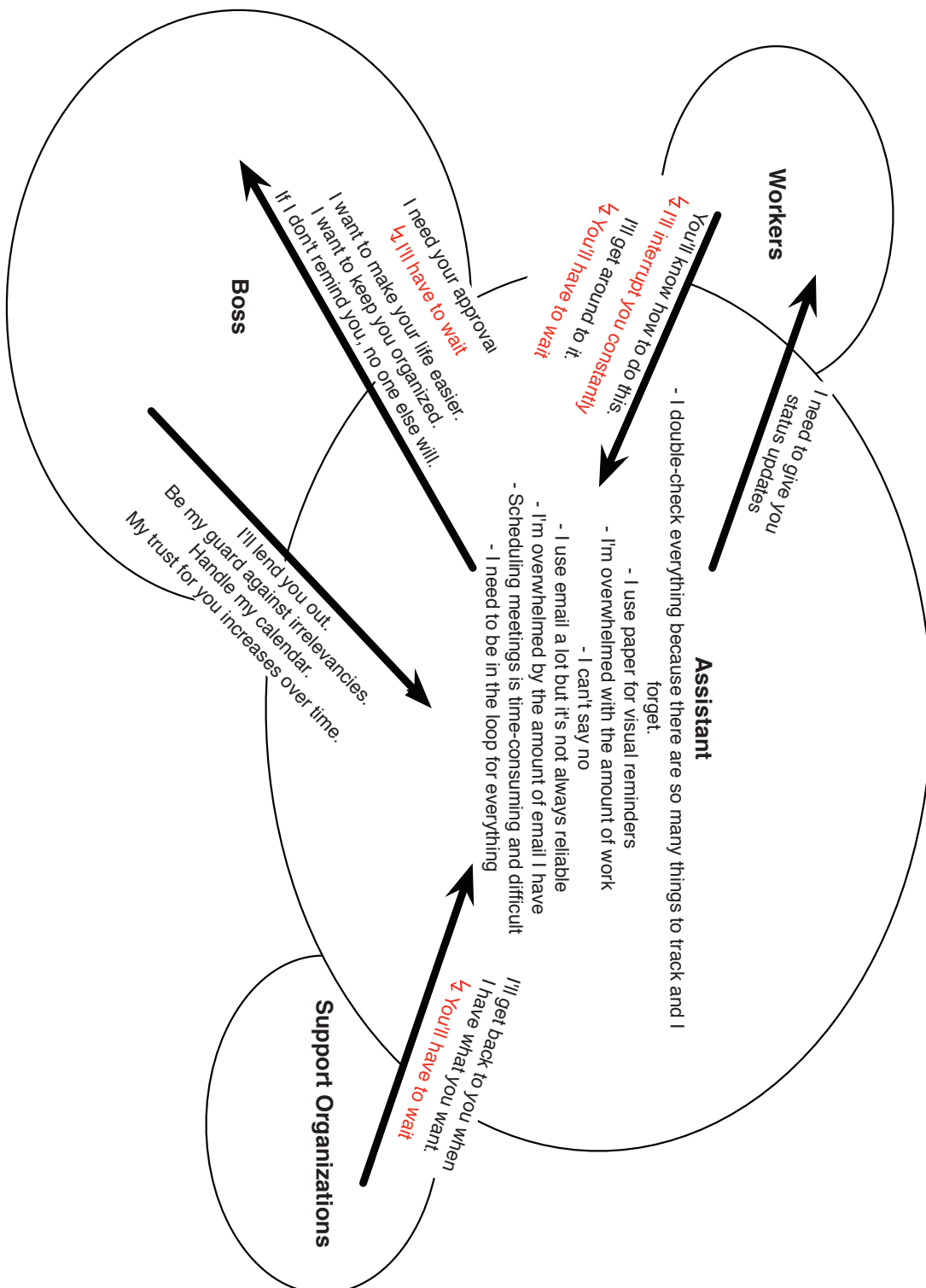


Figure 2

Assistant Physical Model (Consolidated)

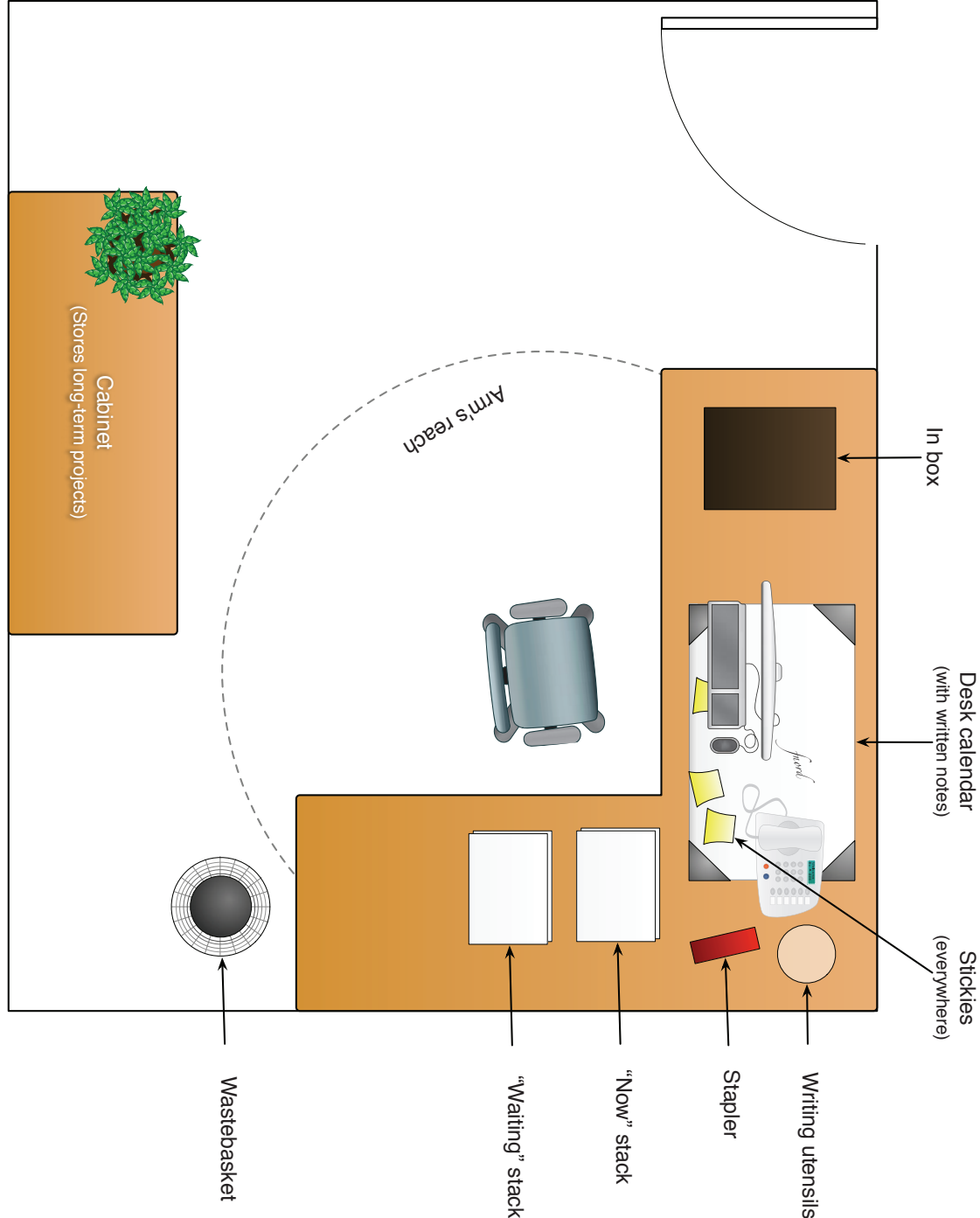


Figure 3

Coordinator Consolidated Flow

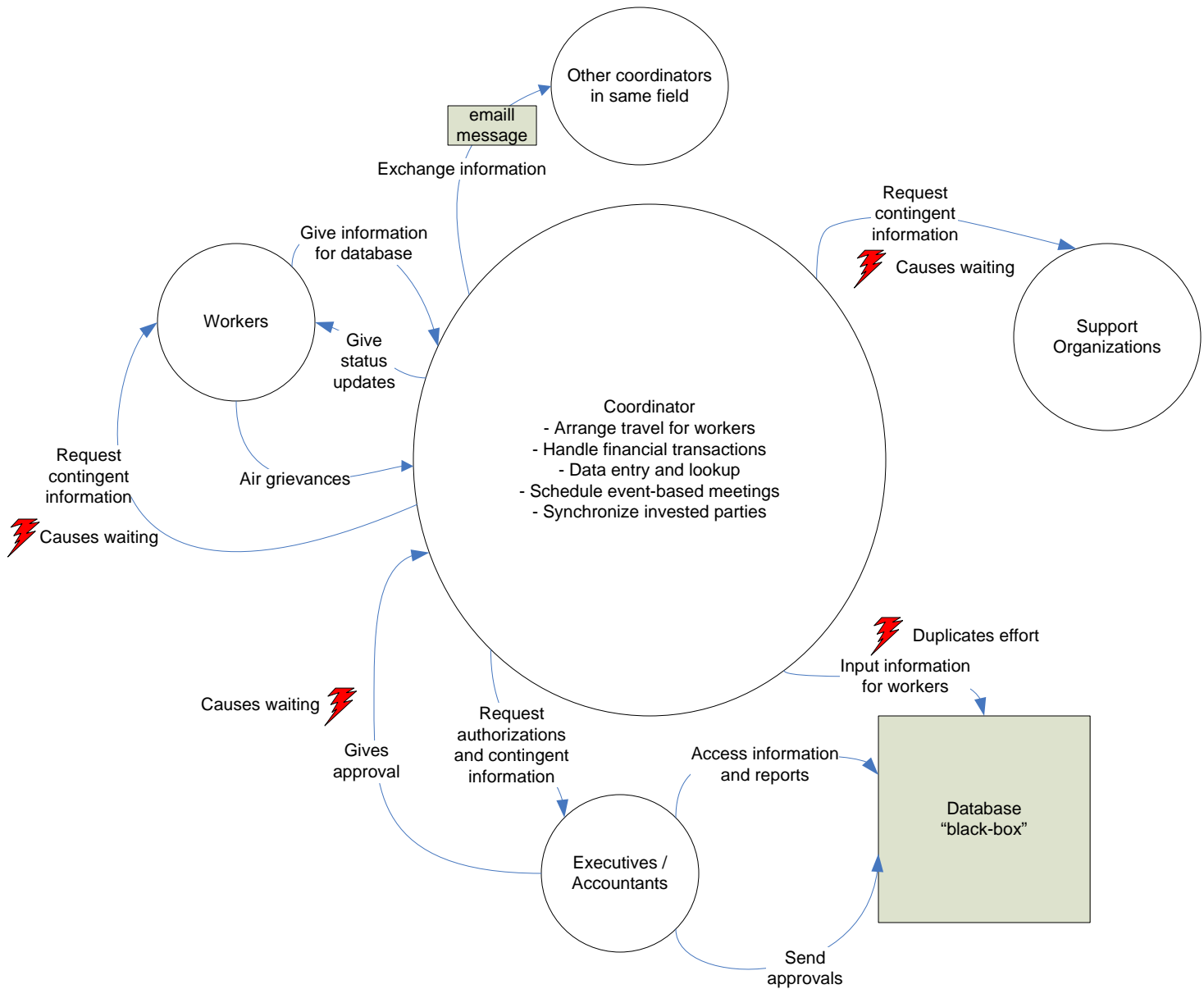


Figure 4

Consolidated Secretary/Coordinator model

Activity	Intent	Abstract Step
Schedule Meeting		Trigger: gets request for meeting
	Determine meeting constraints ex. Time, date, people	Look up or email appropriate people, check boss' schedule
	Ensure venue availability	Check room scheduling database
	Ensure participant availability	Email participants or check schedule if available ⚡ have to wait for responses no centralized database out of date database
	Cancel meeting	Trigger: key participant is unavailable Email participants or remove from calendar if available
Make sure meeting happens	Remind about meeting	Trigger: meeting impending Email participants and walk into boss' office to remind them
Activity	Intent	Abstract Step
Determine travel possibilities		Trigger: someone needs to travel
		Look up event information
	Determine travel constraints (when, how long, where)	online communicate with traveler
	Gather information	Contact travel agent or travel websites
Schedule travel	Negotiate with traveler	Email or present options to traveler
	Request funding	Fill out a form and submit to a database or accountant
	Readjust schedule	reschedule meetings if applicable

Figure 5

	Build itinerary	prepare/consolidate documentation and send to traveler
		Trigger: employee has receipts ⚡ if employee kept receipts
Travel wrap up	Manage reimbursements	Gather receipts, fill out forms and forward to database of accountant
Activity	Intent	Abstract Step
Manage financial transaction		Trigger: gets request for financial transaction (purchase, reimbursement, payment) Receive artifacts that justify the transaction ⚡ artifacts may be incomplete, cause further steps to get all information
	Collect information	Submit a request for authorization through email or database
	Request authorization Notify accountant	Enter into database

would extend to a large number of fields, whereas designing for a specific type of executive has a lower generality.

Executives

Decentralized information. Information applicable to the executive is typically spread across many different repositories, and it exists in many different forms (Fig. 6). Many executives view their assistants as useful for collecting and distilling all information into one form that is easily digestible. CALO seems well suited to collecting information from a diverse number of repositories, so it may prove useful to support visualization of this information.

Buffering work styles. The executives interviewed each had different preferred tools and styles of working. They expect their assistants to act as a buffer between their preferences and those of others with whom they interact. It is more important that assistants learn their executives' styles of work than their actual job description (Fig 7). In fact, we see executives desiring to bring their assistants with them to new jobs for exactly this reason. CALO has this portability—the ability to learn and maintain the executive's preferences is paramount. Learning to handle a large number of data formats is a foreseeable problem that would certainly need to be addressed at some point in the future.

Collaboration is fundamental. As we interviewed higher level executives, we noticed that their jobs get more service-based. Typical work requirements include creating reports and presentations, setting requirements, and reading large amounts of email (Fig. 6). These sorts of activities require communication among parties who are often physically separated. CALO already has some facilities that support collaboration, such as meeting annotations and presentation generation, yet it can be better integrated to support executive work flows.

No common sequences. We recognized that supporting the nature of the executives' work is not as important as supporting the underlying communication between them and other parties. Any non-specific, repeated sequence represents an inefficiency in the executive's work flow because such repetitive tasks should be within the responsibility of their assistants. As such, the sorts of sequences that CALO would have to support for executives would necessarily be domain-specific. Thus, CALO would have to be designed with this domain knowledge in mind, although ultimately it may be possible to make CALO customizable by an expert user.

Consolidated Executive Flow model

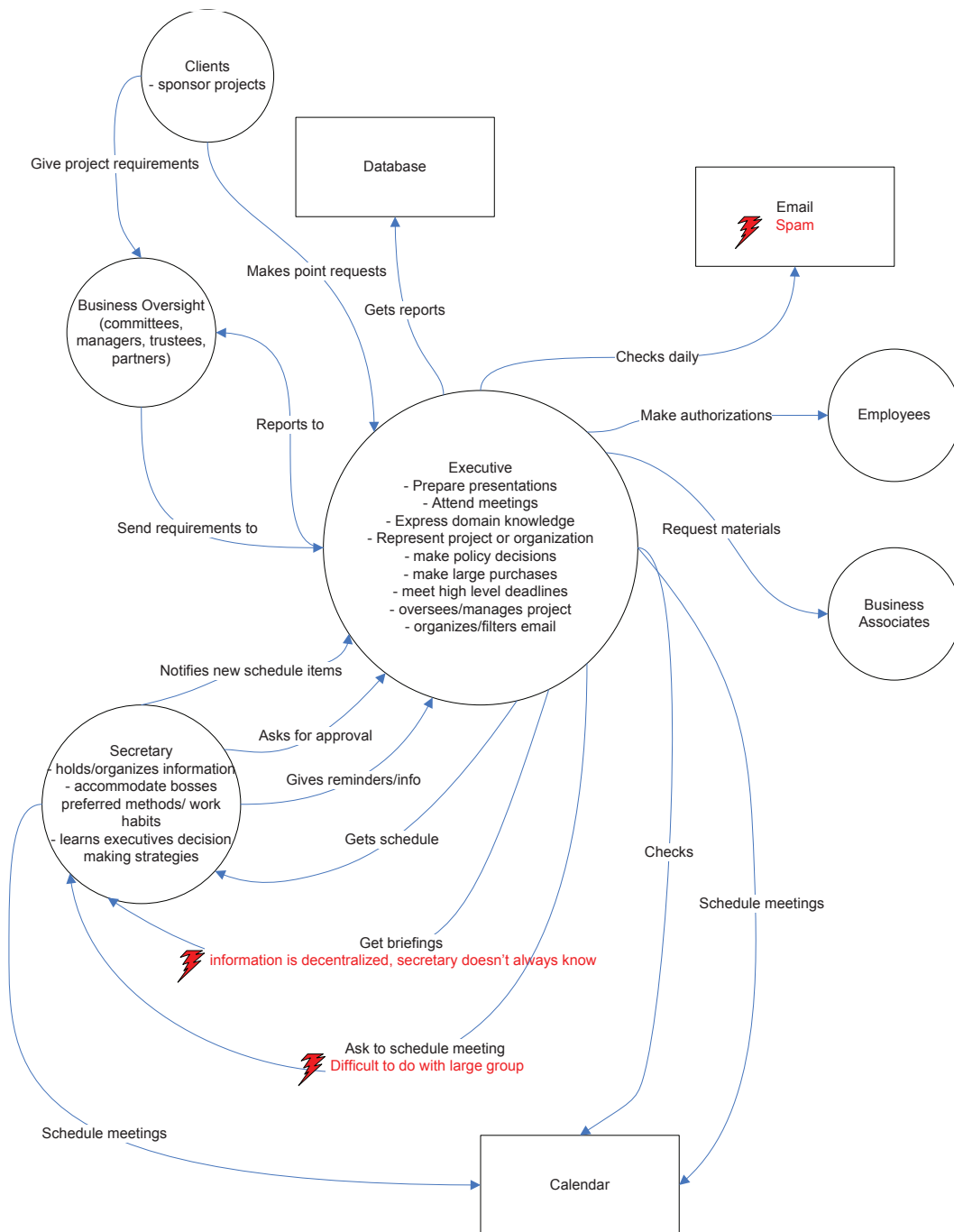


Figure 6

Executive Consolidated Cultural Model

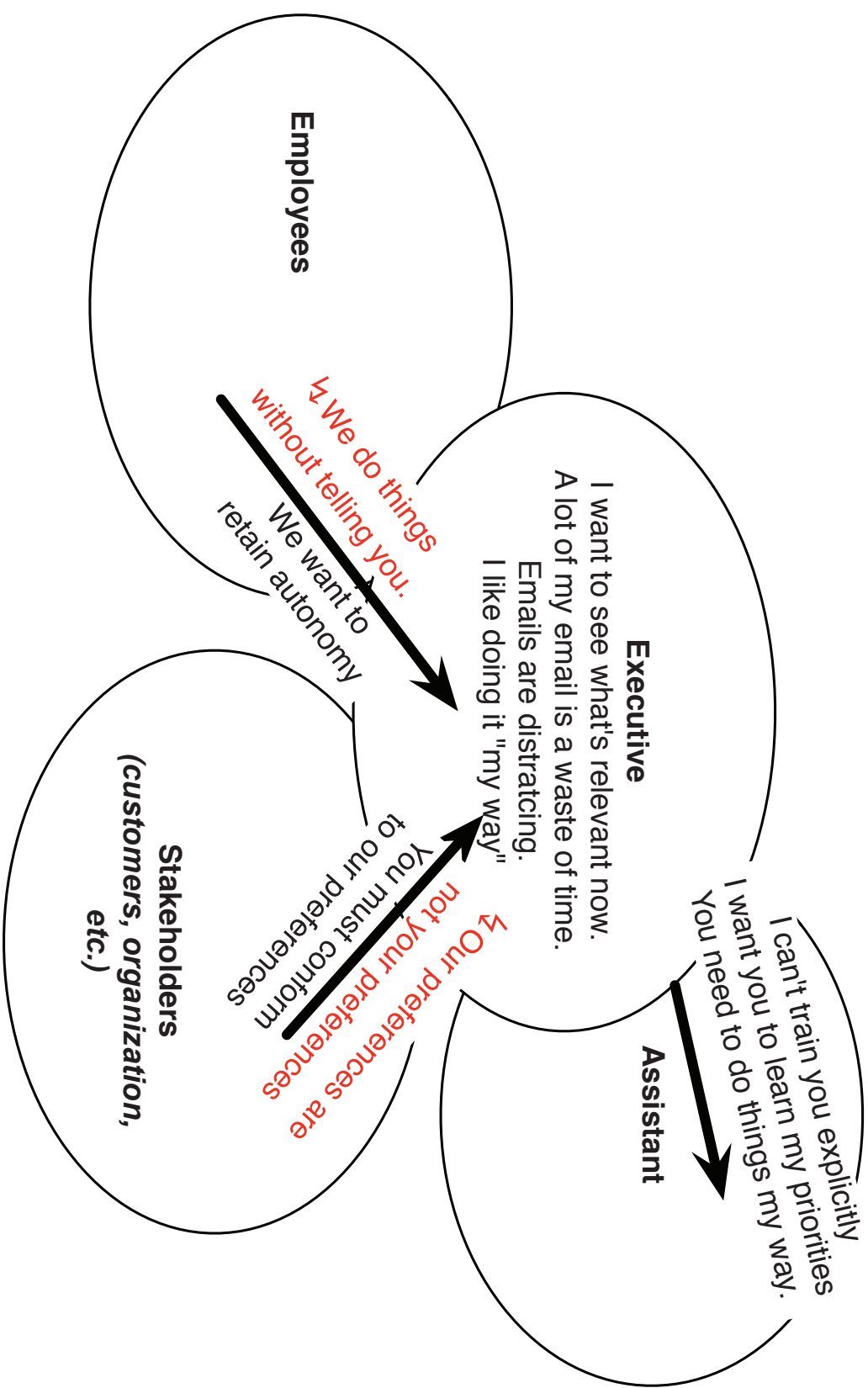


Figure 7

Focus

These insights, taken together, serve to inform and support the direction and focus of our design process. In particular, we see many opportunities based on our research for improving and augmenting the time management portion of CALO. Additionally, we intend to explore ways of bringing together all of CALO's knowledge and learning abilities to support the problems we have identified related to time management. By learning what users are doing at the moment, what they should be doing, and what they will likely be doing in the future, CALO can help users to prioritize the tasks they need to perform, keep track of tasks that may depend upon external factors, such as those that require waiting for other people. This approach will improve the adaptability of CALO's time management features while simultaneously reducing the cognitive load on end users. We also intend to explore ways of having CALO adapt to changing user priorities, and provide non-invasive support and suggestions. Lastly, we plan to search for methods of enhancing, rather than replacing, existing collaborations between executives and their assistants.

c. Personae

Personae are personifications of the user archetypes that our research identified. They exist to present characteristics of the user models in a form that is easier to think about and design for.



Janine

"I can't say no."
"I need to know everything."
"I'm old school."

About Janine

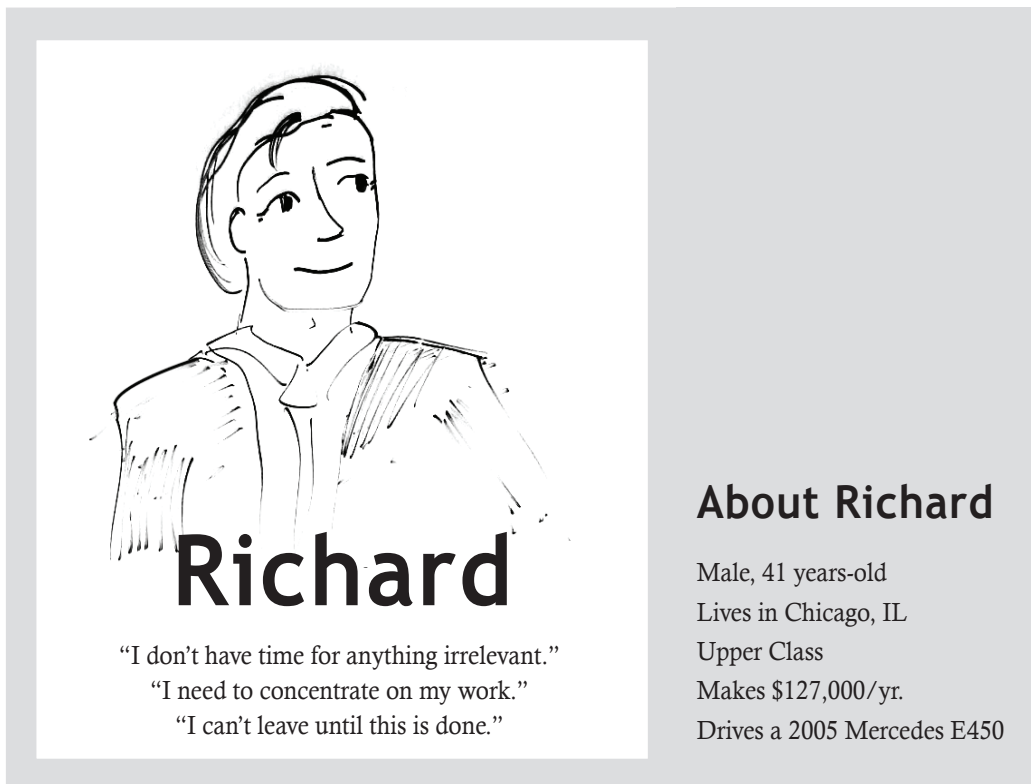
Female, 47 years-old
Lives in Chelsea, MA
Upper Middle Class
Makes \$46,000/yr.
Drives a 2003 Toyota Camry

Job Description

Janine is an administrative assistant at Gaither & Associates, LLP, a medium-sized law firm in Boston, Massachusetts. Her job is to assist her boss in handling travel arrangements, arranging meetings, and handling purchasing. She goes into the office at 8:30am, and usually leaves around 6:30pm, or whenever she finishes all the work that her boss requires her to do for that day. She has worked for her boss for the past 4 years, and sometimes calls him "Bobby."

Life Story

Janine grew up in Atlanta, GA. She attended Agnes Scott College, a national liberal arts college for women. She received an associate's degree in English, and was planning on pursuing a bachelor's degree when she decided to quit school and get married. While she had some side retail jobs at Woolworth during her college years, she became a full-time housewife and mother for the next 15 years. When her two kids were just toddlers, her husband's job was relocated to Boston, so the entire family moved to Massachusetts. When her children entered high school, she decided to re-enter the work force and found a job as an assistant at a law firm. She is not entirely computer literate, but she had great organizational skills and eventually picked up the technical knowledge she needed for her job.



Job Description

Richard is a business consultant for Fantus, LLP, a consulting company that handles corporate site selections in Chicago, Illinois. He often travels to sites, visits client companies, accompanies his clients to the sites, and holds meetings with them and his colleagues. His schedule is often unpredictable, and it requires him to stay in his office until his work is done. He relies heavily on his secretary to arrange his frequent travel.

Life Story

Richard grew up in Toledo, Ohio. He received his MBA degree at the University of Michigan, and his first job was being an assistant supply chain manager at Gillette in Cincinnati. He got married and had one daughter. He accepted a job offer from Fantus, so his entire family relocated to Chicago. After a few years, he and his wife filed for a divorce due to his pressures at work, and now he sees his daughter twice a month. Richard maintains a healthy lifestyle, on top of working 60+ hour weeks.



Job Description

Sharon is a coordinator at Lifehouse Incorporated, a charitable organization that helps people with developmental disabilities. She mainly coordinates payroll, supplies, and travel arrangements, especially when the organization sends employees out to attend conferences. She begins her day at 7:00am, and leaves strictly at 3:00pm so that she can pick up her son from school.

Life Story

Sharon grew up in Santa Cruz, CA, and attended the University of California, Berkeley, where she majored in Economics. At her first job, she worked at a bank as a customer service agent in San Rafael. She got married, and eventually quit her job when she gave birth to her son. She spent the next 5 years as a full-time mother, then picked up a part-time job at Lifehouse when her son entered pre-school. She started off as a receptionist, and after a couple of years, she became a full-time coordinator for the organization. Sharon is very devoted to her work, and also to her family. Sometimes she has to bring her son into the office with her, because there simply is not enough time for her to complete her work and take care of her family at the same time.

d. Use case analysis

A use case is a general scenario of how a particular sort of user might use a software system. Use cases are valuable for their ability to reveal system requirements and user roles, and also for solidifying nebulous foci into well-defined system capabilities. Since collaboration is among our foci, all three types of users: secretaries, coordinators, and executives appear in our use case diagram (Fig. 8). Some of the use cases with which these users are involved and which fall within our focus area include scheduling and being reminded about meetings, adding to-do items and being notified about them, and interacting with CALO's perception of the user's priorities; all of these will be priorities for prototyping.

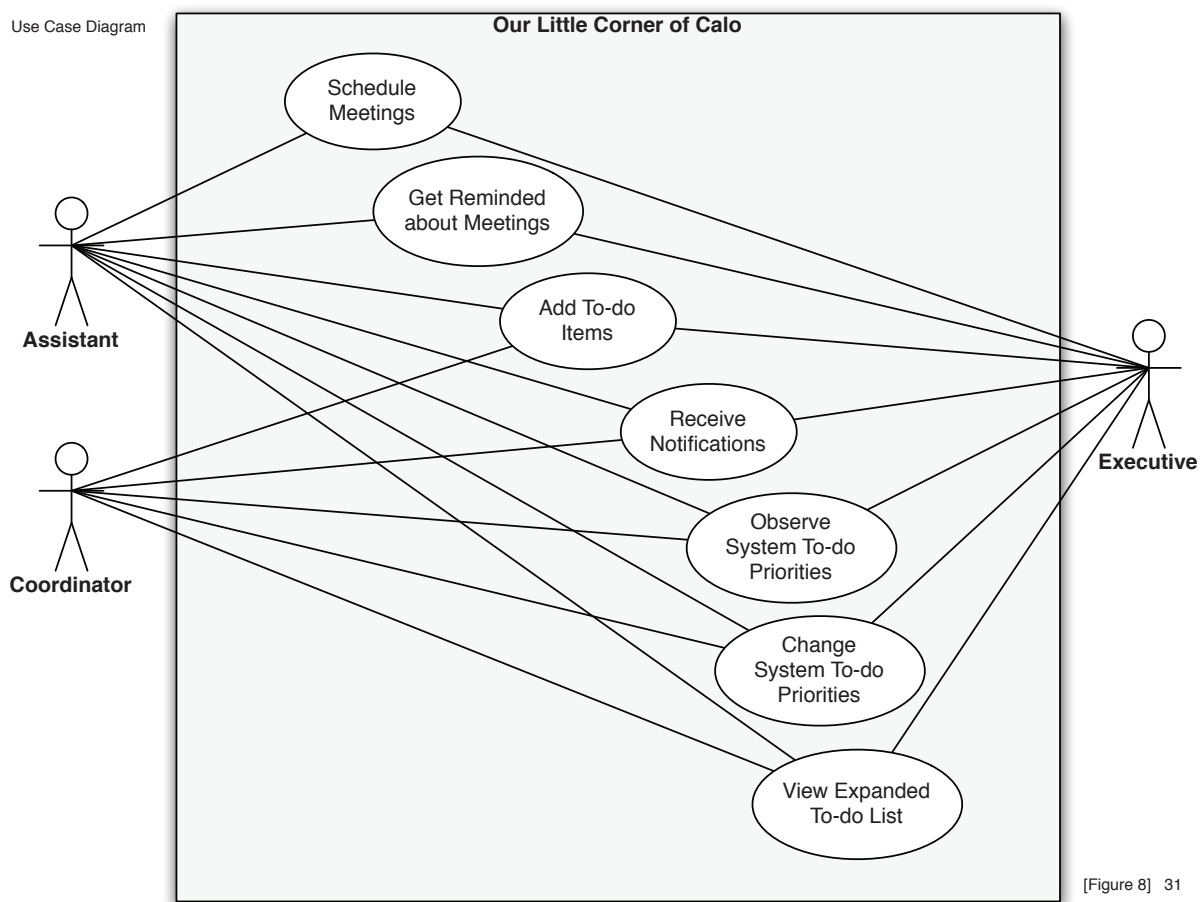


Figure 11

2. Ideation and design

a. Brainstorming: initial ideas

After gathering data regarding users' work flows, work habits, and their tools, the CALO *Stardust* team made an affinity diagram of design ideas that could potentially address the needs and support the breakdowns of our target users. This brainstorming directed us to two broad directions for the project: one, to design a system that supports the user's task management, and two, to design a way for CALO to support the collaboration between CALO users (particularly executives and their assistants and subordinates). Task management would include components that handle task prioritization, notifications, task visualization, activity documentation, and task organization. User collaboration, on the other hand, would support components that allow users to schedule meetings, task delegation, and other explicit forms of collaborative communications.

We attempted to clarify the design ideas from the affinity diagram so that they could work together in a larger, coherent system. Our team realized from brainstorming various ideas and concepts that at any time, the user needs to have certain information visible at all times and information available by request. We also noted that the ability to modify the system would be important so that users would not feel commanded by the system. It also became apparent that having these components in a sidebar on the desktop was the best way to present them to the users. With this in mind, we individually and as a group we came up with several ideas before choosing the best to pursue.

The team saw that a sidebar would be a space for quick, at-a-glance information while information requested by the user, would have in-depth detail about the items in

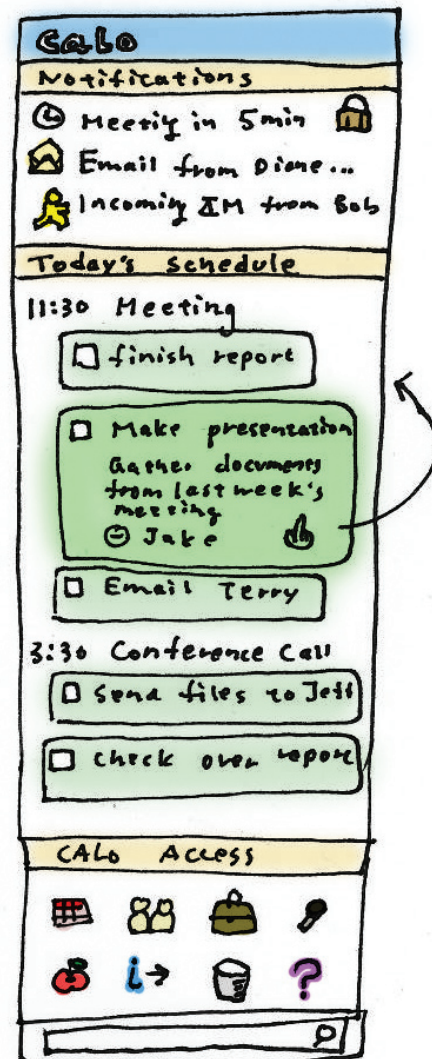


Figure 12

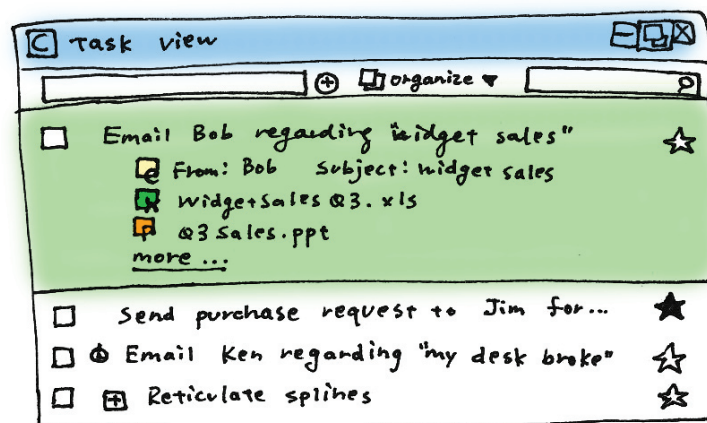


Figure 13

their sidebar. In response, we formed the idea of the task viewer which would allow the user to access a complete set of information about his or her tasks while the sidebar would only show the important details of the task. More detail about the task viewer is explained in the features section.

We adapted the concept of the PrepPAK from the current CALO system and reimagined its role as a general store of resources pertaining to a task, an event, a meeting, or any other such piece of information. The pack window was designed to closely resemble a folder in Windows Explorer to provide the user with a familiar interface in which they could exert full control over the contents of the pack, including additions, deletions, renames, and other operations of that sort.

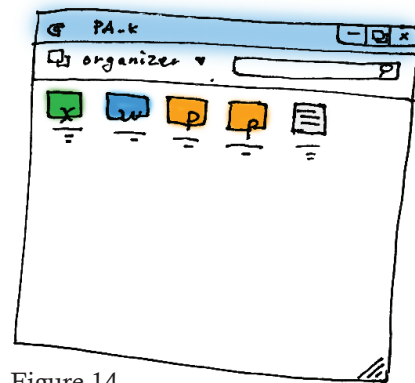


Figure 14

We realized that executives tend to observe their work on a higher level, especially between different areas of work. Therefore, we created task visualization ideas to help see an overview of the user's work. One task visualization called project navigator, shows tasks grouped by the projects they belong to where the importance of the project is reflected in the size of the object. The location of each project would depend on their priority with the center being most urgent. Each project box would show tags, notifications, a few urgent tasks, and obviously the smaller the project appears, the less of these details would show and therefore do not need as much attention. In the project navigator users would be able to see very generally what they need to attend to right now, and at the same time, get an overall sense of what other projects they are neglecting.

A similar concept, called the "task relationship cloud," creates a cloud of tasks where the tasks' size relates to its importance, and the location refers to its priority. However, the task relationship cloud would only show tasks instead of the projects they belong to. The distances between them would indicate the levels of their association and arrows would show the tasks' dependencies. For instance, if one task could not be completed until another task is done, then the task relationship would

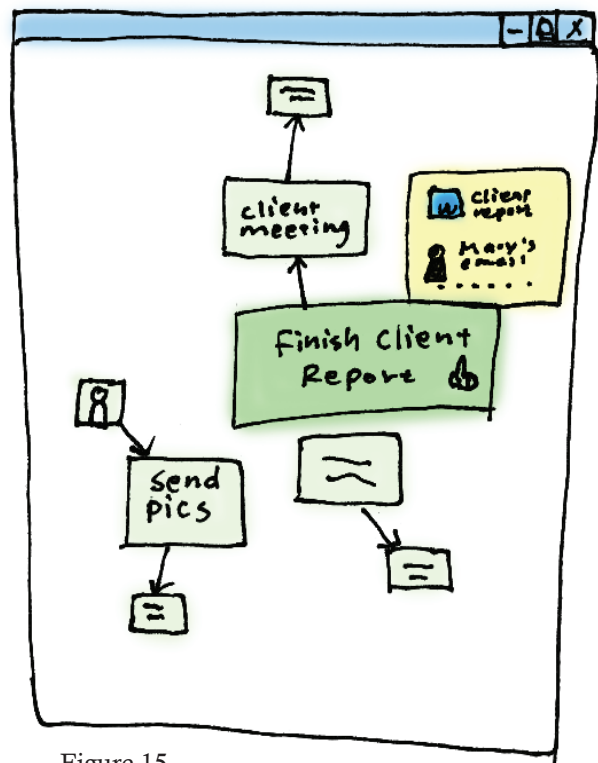


Figure 15

visualized the chain of events that must happen before a task can be started.

A third, slightly different, way of visualizing tasks was what we called the “Spiral,” where the most urgent tasks would be placed in the outermost ring of a “spiral of tasks,” and the least urgent tasks in the inner curves of the spiral. Also, the size of the tasks would gradually get bigger (hence showing more details) the more urgent they are. In the spiral, the user can easily gauge how many more tasks are left in their workday, and when they see the end of the spiral, then it means the user is almost done with his or her tasks. It was an unconventional way of representing tasks, and our team had jokingly called it the “spiral of despair.”

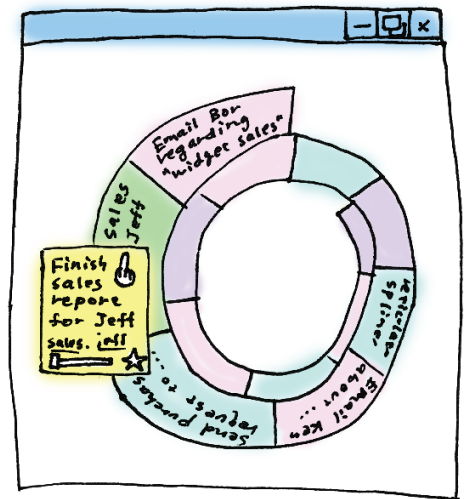


Figure 16

After lengthy discussions, our team decided to focus more on the task management direction and less on collaboration, simply because of time constraints for our project, and also because we had more ideas and interest in this particular topic. We crafted certain requirements that our new design should fulfill, and a list of task management features that could support these requirements. Narrowing down our initial ideas, we agreed that the basic design requirements were to allow the user to change priorities, see relationships between tasks (and task details), manage their time, receive notifications, and access to both user and system activity documentation. The ideas were eliminated or merged into general feature categories: project-based priorities, task relationship cloud, the “spiral,” learning log, dynamic schedule, notification center, notification modalities, (detailed) task viewer, project navigator, and a to-do list on the sidebar.

	Project-based Priorities (S)	Task	Spiral of Despair (S/R)	Learning Log (R)	Dynamic schedule (S/R)			Task View (R)	Project Navigator (R)	To-do's on side bar (S)
Changing priorities		weak	strong		weak	weak		strong	weak	strong
Seeing relationships (between tasks)		strong	weak		weak			weak	strong	
Getting context (ie: what project is this for? who is involved? anything not tasks)	weak			strong		weak	strong			weak
Time			weak		strong					
		weak		strong	weak			weak	weak	weak
Notification	weak	weak	weak		strong	strong	strong			strong

Figure 17

b. Concept validation

Concept Validation is a method used after the ideation phase to confirm whether the ideas generated from the research reflect the users' actual needs. The method involves showing target users storyboards of situations identified in the research and solutions developed from the ideation phase. We created the following storyboards to test ideas of time management, collaboration, information management, system automation, training and others. Concepts were validated on four executives and three administrative assistants, and the following is their feedback:

What should I do now?

Figure 18



In this scenario, the screen displays what the system thinks is the best task for the user to do next. Executives thought that a sort by priority would be useful; however, giving options was seen as preferable to presenting a top choice. Coordinators or assistants were not as receptive as executives because they did not appreciate the idea of the system telling them what to do. The coordinators felt that they were quite capable of prioritizing tasks and did not need a system to do it for them. However, giving several options was acceptable because it made the user feel like he or she was the one choosing the task rather than the computer.

Suggestions for Better Time-management

Figure 19



This scenario tests the idea of helping users get tasks done in the remaining time available without spending extra time and effort on deciding what to work on. Executives did not readily identify with this need since they did not want their free time “filled in” with tasks by the system, and estimating duration of a task is difficult making the accuracy questionable. Some coordinators however were more open to the idea of estimating duration possibly because their tasks are similar or more repetitive. Overall, coordinators accepted the scenario as long as they still maintained control over the system.

Seeing resources associated with a task

Figure 20



This storyboard demonstrates the ability to collect all the documents or resources related to a conference, trip, meeting or task in one place. All of our executive and coordinators agreed that the ability to gather relevant resources easily would be very useful for them.

Receiving reminders

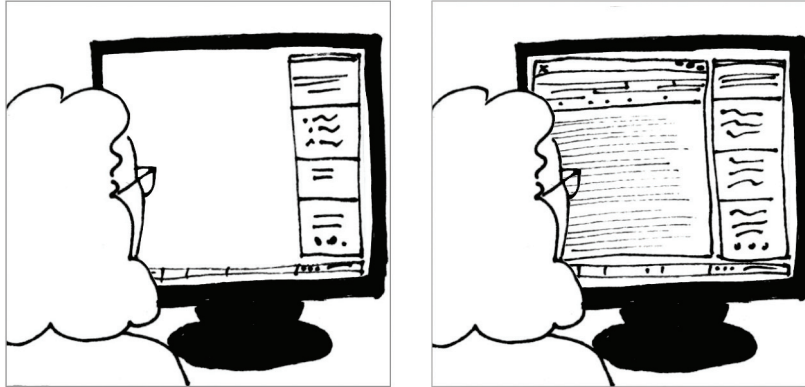
Figure 21



For this concept, we wanted to see if users would like to be notified of important or timely events. Among executives, we found that many liked the idea of being reminded but had varying degrees of intrusion that they were willing to accept. Coordinators had mixed reactions toward reminders based on personal preferences and work styles.

Information at your finger tips

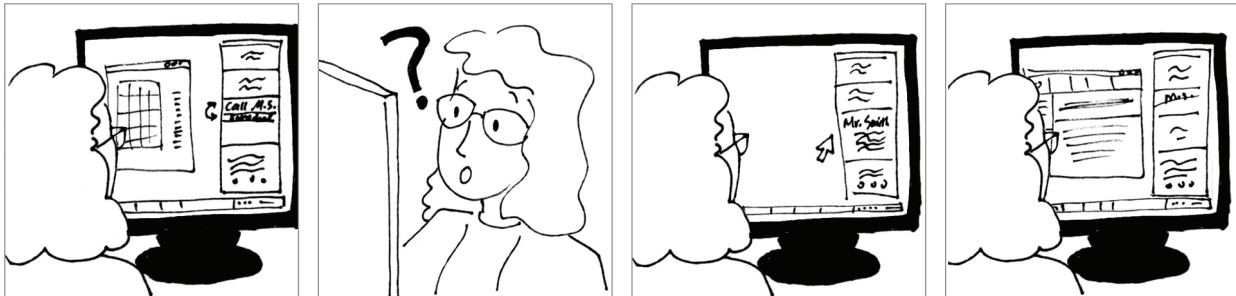
Figure 22



This is a simple interface concept of placing a sidebar that cannot be covered by other windows, on the users screen at all times. Both coordinators and executives were not enthusiastic about having a sidebar visible at all times. Users did not think there was much information worth being on the screen at all times and taking up valuable screen real estate. However, they were willing to use a minimizable or smaller version of a sidebar to access information easily and hence our creation of the minibar later on.

Access to the system's reasoning

Figure 23



This concept tests the idea of having access to CALO's artificial intelligence reasoning. Executives thought that explicit AI reasoning like it's statistics analysis would not be very helpful. However contextual information, like what document or piece of information the AI based its decision on would be enough to understand its 'thought process.' Coordinators had a similar response however one also answered that she would not have time to deal with things such as the checking the AI's reasoning even if it did something she did not understand.

Getting back up to speed

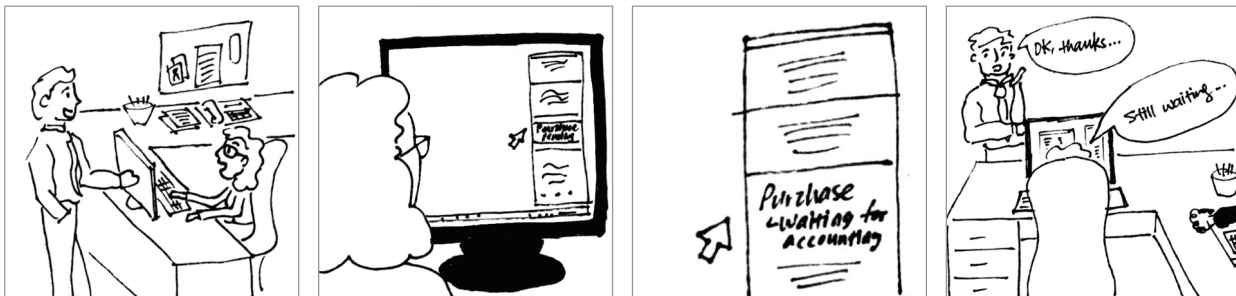
Figure 24



This scenario tests the idea of recovering from an interruption. Most of the executives agreed with this concept, but noted that getting the right level of granularity would be crucial. The information presented must be more precise than “what task was I working on,” but less precise than “what characters did I just type”. Coordinators did not see interruptions as detrimental to their work and therefore did not identify as much with this need.

Seeing pending tasks

Figure 25



The concept of pending tasks was identified in our contextual inquiries as when users were unable to continue working on a task until they received something from a colleague. In this scenario, the user is able to check why she hasn't filed an expense report and the system shows her she is still waiting on a package slip from her supplier. Both executives and coordinators responded that the idea of pending tasks happens frequently and that it would be helpful to easily keep track of them.

Taking time to train the system

Figure 26



CALO is a system with a sophisticated artificial intelligence that can accomplish more with training. This scenario shows how explicitly training the system would make it learn quicker. An alternative scenario of the same concept had implicit training where the user did not tell the system rules or preferences. Instead, the system learned from the users' actions and inferred its reasonings. Executives and coordinators preferred implicit rather than explicit training. Users thought it would be difficult to explain the reasoning to a system and rules would become too complicated. A few were willing to train while setting up the system, but training was still seen as intimidating and the user might not know what preferences he or she would want yet.

Letting the system take over your task

Figure 27



In this concept, we were validating whether users would want the system to automate tasks for them, and to what degree of control over the automation they would want. Overall, both user groups found automation to be useful as long as there was the opportunity for user input along the way since few tasks are 100% the same. However, interestingly, one executive intended to use automation as more of a documentation tool for infrequent tasks to remind her of the process. Another executive answered that they did not have enough repetitive tasks to be automated.

Seeing the progress of work

Figure 28



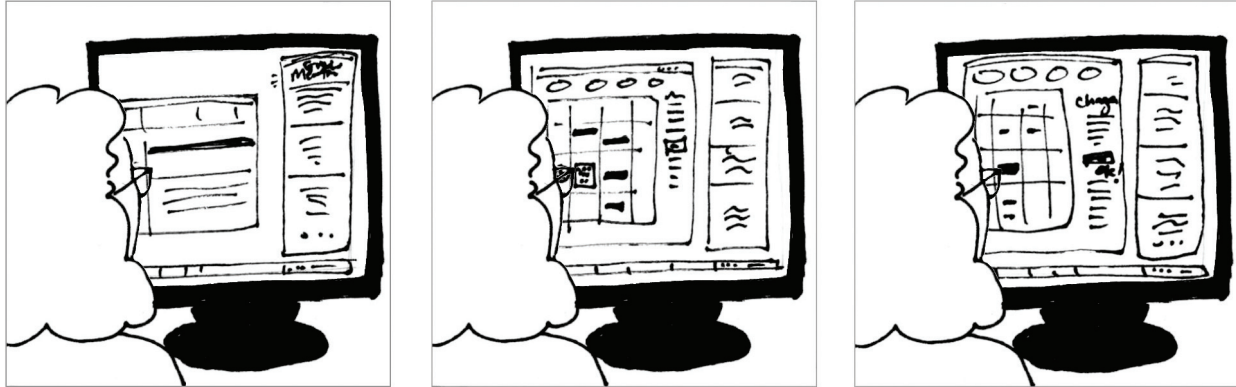
In seeing the progress of work, the system keeps track of how much has been completed for the user and what work is left. Executives did not see this feature as useful because their work is complicated and they didn't need that level of detail. Coordinators had a similar reaction where they did not see a need for it.

Capturing loose items to do

Figure 29



Here we were trying to validate the concept of the system automatically documenting small tasks that cannot be completed at the moment and are traditionally written on post-its, scraps of paper or just kept in memory. Executives and coordinators found this very useful since it is easy to forget small tasks you can't get to right now.



This scenario presents the idea of allowing select people to be able to view and edit a user's calendar. The user would be able to see all changes and confirm to approve them. Although both user groups found this idea to be useful, it has been implemented by several calendaring agents already with varying levels of success. Success usually depends on having the entire organization using the same calendaring tool consistently.

3. Evaluative user testing

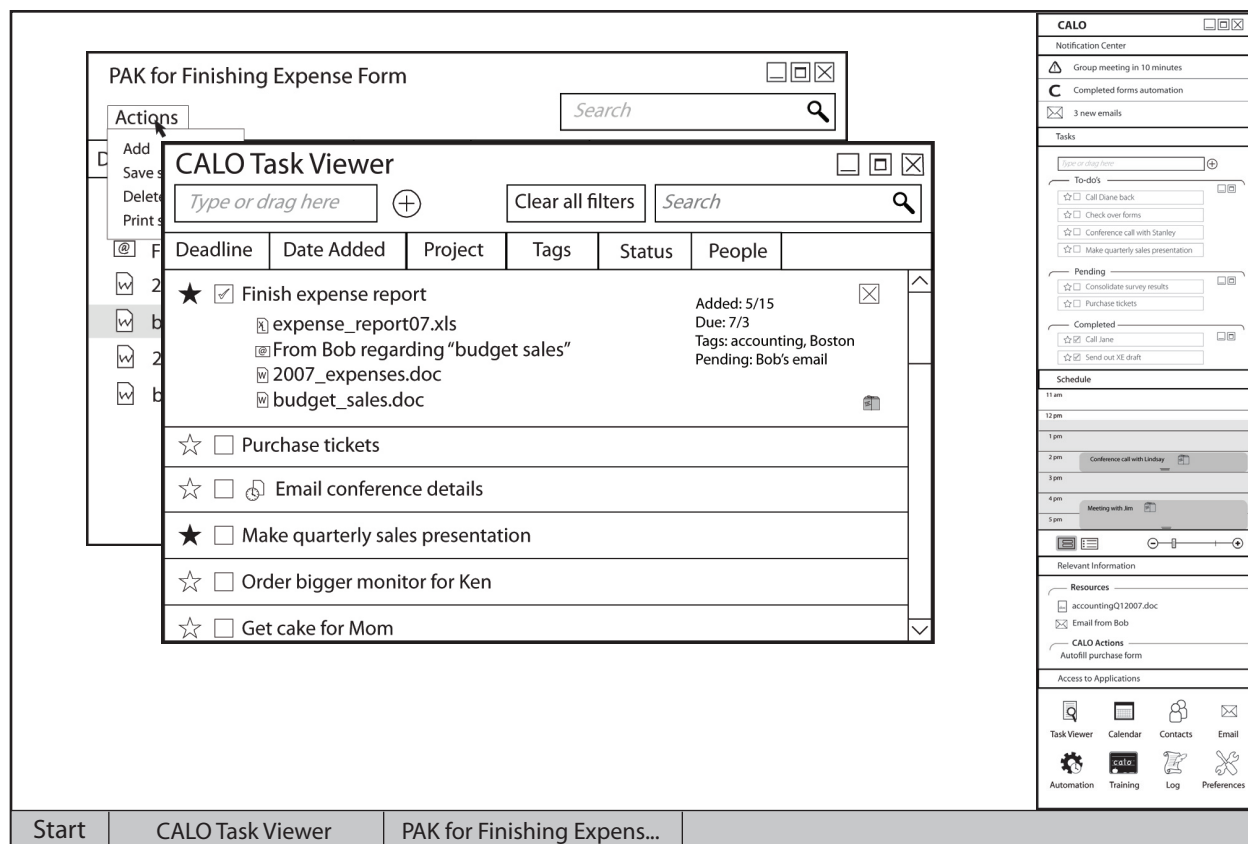
a. Think-alouds with paper prototypes

Gathering all the results from concept validation, our team started to work out the details of our design—how to integrate all the different components together, how the sidebar interface looks, and how users would interact with different parts of the system. We decided that within the sidebar there would be five panes: notification center, task list, schedule, relevant information, and access to applications.

Once we created paper prototypes of these panes, we ran our first round of think-aloud user studies on non-target users. We used non-target users to test for usability issues and for any conceptual inconsistencies. The user test gave users a background story in which they were an executive in a large company. The study included a list of tasks that involved editing tasks, checking their schedules, receiving notifications, and finding resources. Below are the results from our first and second round of think-alouds on non-target users. From these results we drafted a few changes to our paper prototypes to do a third and fourth round of implementation user tests.

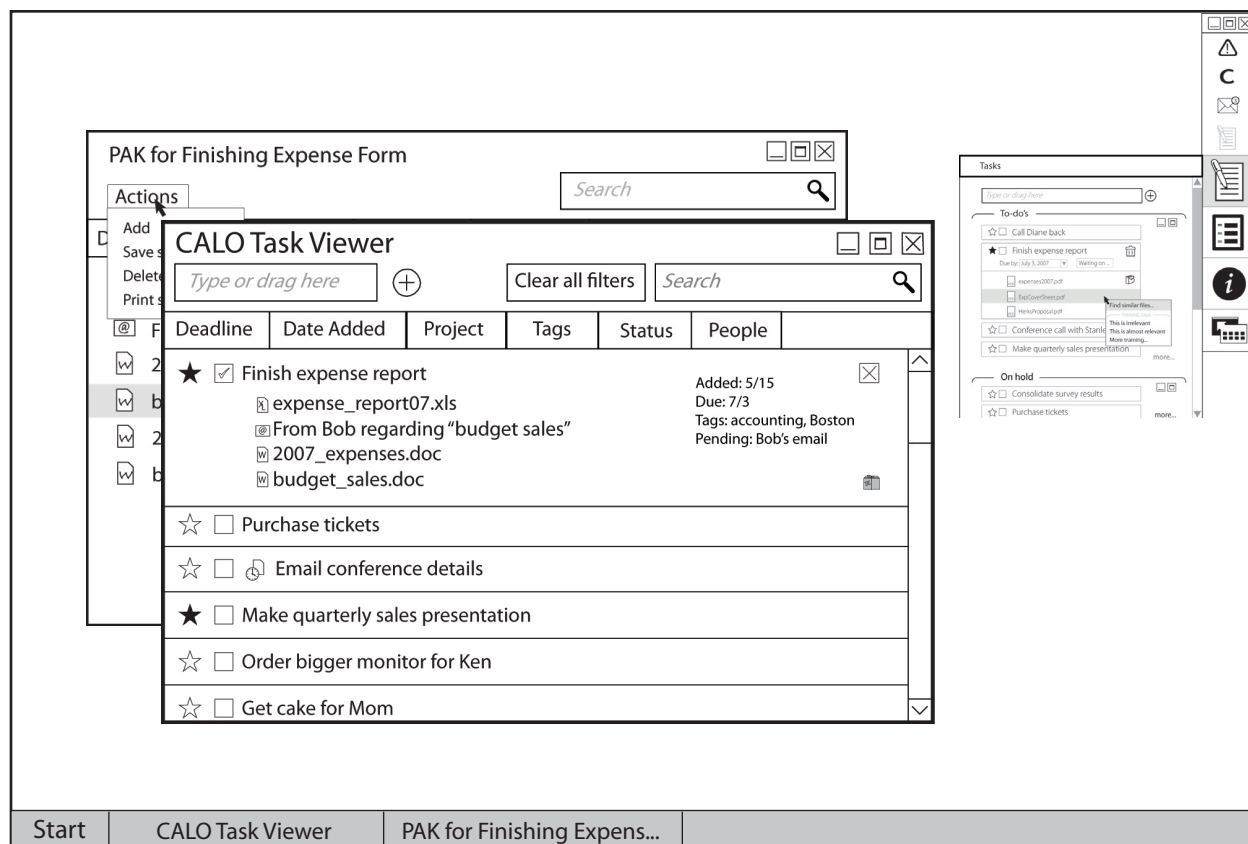
Think-aloud Study I: Sample Screen (used the normal sidebar)

Figure 31



Think-aloud Study II: Sample Screen (used the mini-sidebar)

Figure 32



Think-Aloud Study I Results

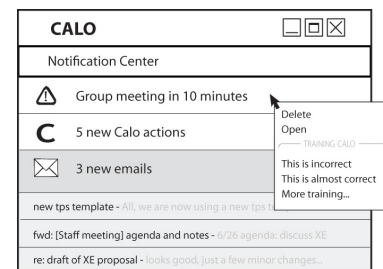
Users:

	Years of Computer Use	Work mainly on Computer	Use digital calendar or task manager
User 1	12	Yes	No, but familiar with Google Calendar
User 2	“Forever”	Yes	Google Calendar
User 3	A few years	No	No

Tasks Users Completed Without Major Problems

1. Notification Center: Handling incoming emails

- Opening the grouped category in notification pane for incoming emails (e.g., 3 new emails) and seeing the details
- Reading the full email by double clicking on the notification to open up the email from the inbox



2. Notification Center: Dismissing a notification

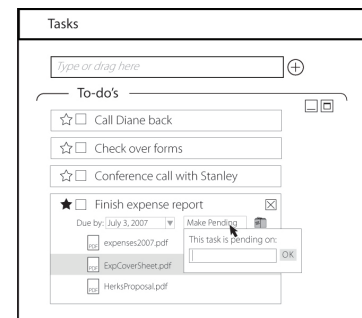
- Clearing the notification by clicking the “x” was easy to understand

3. Task Pane: Adding a new task from the text field

- Users understood the concept of hitting “enter” or clicking on the add icon

4. Task Pane: Marking a task as pending and making a note on what it’s pending on

- There was some confusion about what “pending” meant
- Solution: Change to “On Hold” (still testing)



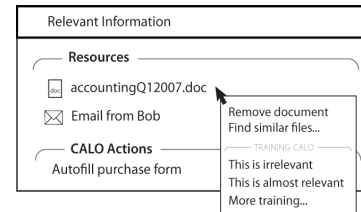
5. Task pane: Minimizing & expanding groups

6. Relevant information pane: Correcting a misplaced file

- Surprising 2/3 users figured out to right click and train CALO rather easily; one tried to drag it out of the pane, which we also see as a valid action and are considering to implement

7. Task Viewer: Filtering and clearing filters

- One person had trouble clearing the filter but maybe he misunderstood the task description; other than that, it seemed intuitive



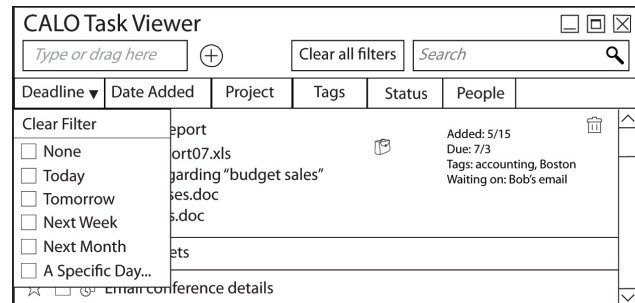
Problematic Areas & Tentative Solutions:

1. Schedule pane: Adding a new event from a block view

- Most users didn't understand that they had to go the agenda view in order to add an event that does not occur on the current day

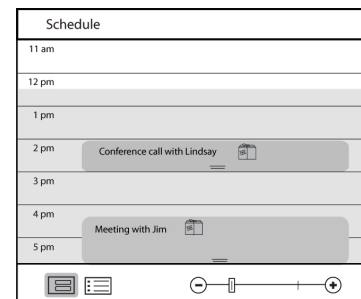
- It was unclear for some users how to get to the agenda view; user would try to use the zoom bar to get to the next day

- Solution: Add an "add" button on both the block and the agenda views; make the zoom so that when users zoom out enough block view is not legible, it changes to the agenda view automatically; similarly, when they are in the agenda view and want to go back to the block view, they could zoom in or click the block view icon (support multiple ways of switching between the views)



2. Task Pane: Finding files to work on

- All users went to the relevant information pane instead of opening the file from the expanded task pane; this is an acceptable work around but still not optimal considering they didn't really understand what the purpose of the relevant information pane

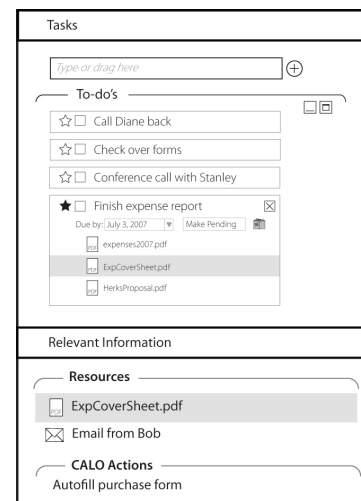


3. Task Pane: Grouping tasks

- Users didn't understand they could drag things around (could be a paper prototype issue)

- There was some confusion of what grouped task is: sub tasks or just grouped?

- Solution: Present clear affordance that tasks are draggable with changed cursor tip to a cursor hand (later on the digital prototype); also, the group should provide the option of adding a title for the grouped tasks



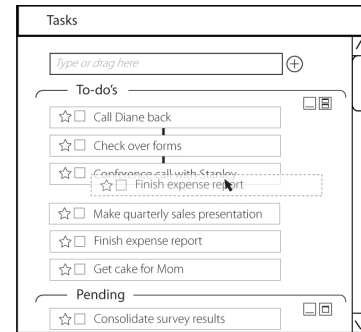
4. Relevant Information Pane: Finding the automation function (when filling out a form)

- Most users didn't think to look at the relevant information pane; some went to application pane at the bottom after they were unable to find it in other places; this is a secondary way to accomplishing this task but still not as intuitive

- Solution: Since “relevant information” is not a very helpful name (what does it mean to be relevant?) we changed it to “CALO Suggestion” pane to indicate the pane contains things that CALO can do for the user; also CALO can inform users that there are actions that CALO can automate for the user

5. Task Viewer: Tagging a task

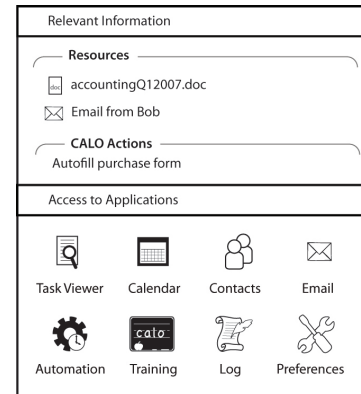
- Users wanted to do this action on the sidebar (only possible in the task viewer; some had trouble finding the task viewer because they didn’t think such a thing existed)



Open-ended Comments from Users:

1. Icons in the application pane didn’t make sense

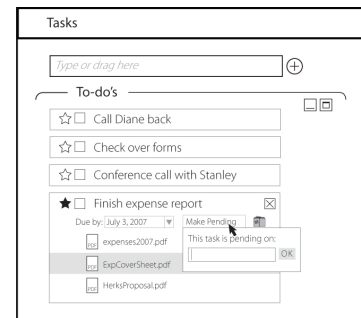
- Solution: provide tool tips and possibly labels for icons
- Didn’t understand there are more tasks when task categories are minimized or only showing a few tasks
- Solution: have “show all N’s...” at the bottom of the list when it’s only showing a few



2. Not clear how to get to the task viewer

Other Changes we made after Think-aloud I

1. Confusion over closing the expanded task and deleting an item
 - E.g., when tasks show details, “x” seems to indicate “close” instead of deleting the task
 - Solution: use a trash can icon for deleting and X for closing



2. CALO Suggestion Pane

- Change the name from “Relevant Information” pane to “CALO Suggestions” pane
- It should show files and actions associated with the focused window
- When it’s focused on a pane in a sidebar, show instructions of how to use that particular pane on the sidebar

Think-Aloud Study II Results

Users:

	Years of Computer Use	Work mainly on Computer	Use digital calendar or task manager
User 4	12	Yes	Google Calendar and Wikis
User 5	15	Yes	Google Calendar
User 6	18	No	No
User 7	16	Yes	No
User 8	15	Yes	No

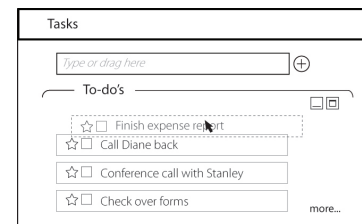
Think-aloud II used the mini-bar version of CALO. All panes appear in the same way except for the notification center.

Tasks Users Completed Without Major Problems

1. Notification Center: Handling incoming emails (same as Think-aloud I)

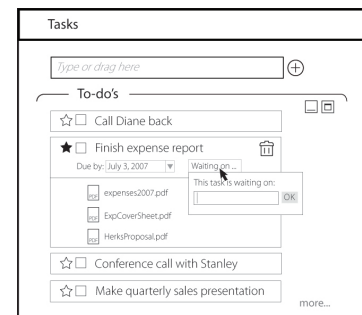
2. Task Pane: Moving a task to change its priority

- Dragging seems to be alright now after users switched from using a pen to a real mouse with a paper pointer



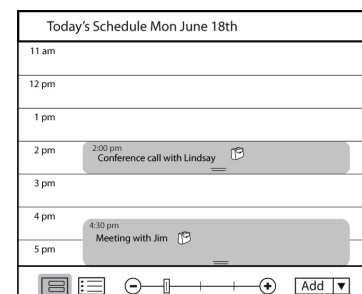
3. Task Pane: Marking a task as pending and adding a note for the reason

- Finding the automation (filling out a form) from the relevant information pane
- Surprisingly alright, changing the name of the pane possibly helped
- Some expected the automation to appear on the browser (or on the file) they are working on



4. Schedule Pane: Gathering resources for an event (e.g., meeting)

- The new pack icon seems more intuitive
- One user had a lot of trouble (perhaps he didn't see the icon)



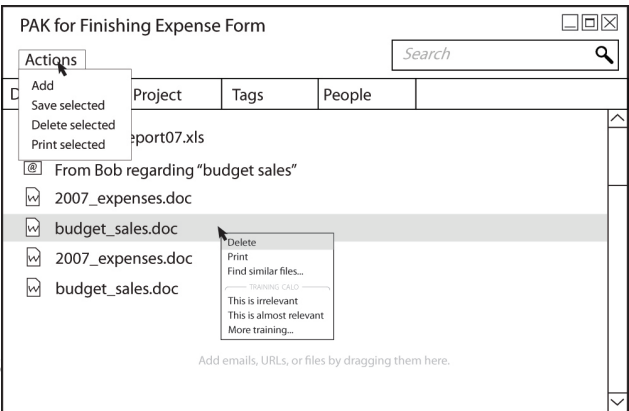
5. PAK Window: Removing and adding files

6. PAK Window: Finding similar files from right-clicking

7. Task Viewer: Tagging

- Works well except for User 8 (didn't understand tagging; perhaps we should include motivation in the task script)

8. Task Viewer: Filtering and clearing filters



Problematic Areas & Tentative Solutions

1. Notification Center: Dismissing irrelevant notification from the minimized sidebar

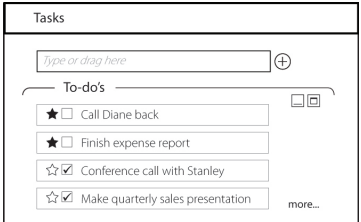
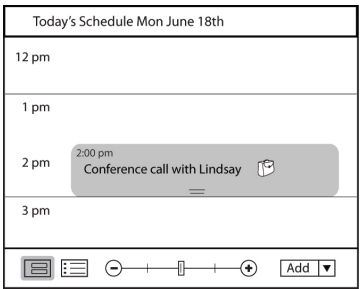
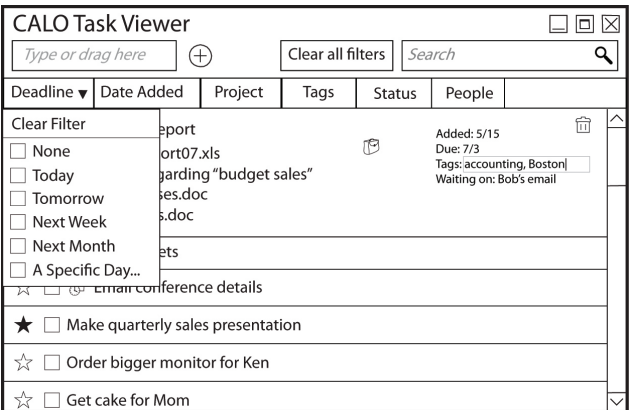
- Almost all considered “Growl” going away as dismissing (“Growl” is a notification application system; for more information, please refer to <http://growl.info>)

2. Schedule Pane: Using the zoom bar to go to the next day

- Some zoomed in instead of out to go the next day

3. Task Pane: Grouping tasks

- Some considered starring tasks as grouping; some would check the completion boxes thinking they are checkboxes



Open-ended Comments from Users

1. Wanted to see more direct actions inside task viewer and PAK window
2. Spelling of PAK is confusing; “Sounds like it stand for something”
3. Task viewer should be accessible from a right click in a task pane
4. The mix of desktop and web-based metaphors (star & checkbox) is confusing
5. Completion box seems too much like checkbox
6. Solution: make completion boxes look more like buttons

Our Comments

1. Putting a new label to indicate that there are more tasks to view if the user expands the group seemed to have helped a lot

Issues After Think-aloud II

1. Users would not notice the suggested actions CALO can automate when the sidebar is minimized
 - Solution: “Growl” when there are actions that can be automated
2. What is the difference between “remove” and “this is irrelevant” in the contextual menu? Former does not train, the latter does. Is it necessary to provide this differentiation? Would users know the difference?
 - Occasions for users wanting to remove files without training CALO: in the case that users get information from outside of the computer (which CALO cannot track), training would be “confusing” to CALO because it would try to make assumptions out of existing information IN the computer; it may be inefficient and possibly harmful to un-train CALO
3. We need to elaborate the interactions from the minimized sidebar more (e.g., what would happen when they double-click on the icons?)
4. We need to elaborate the interfaces more for stand-alone windows (e.g., PAK and Task Viewer)

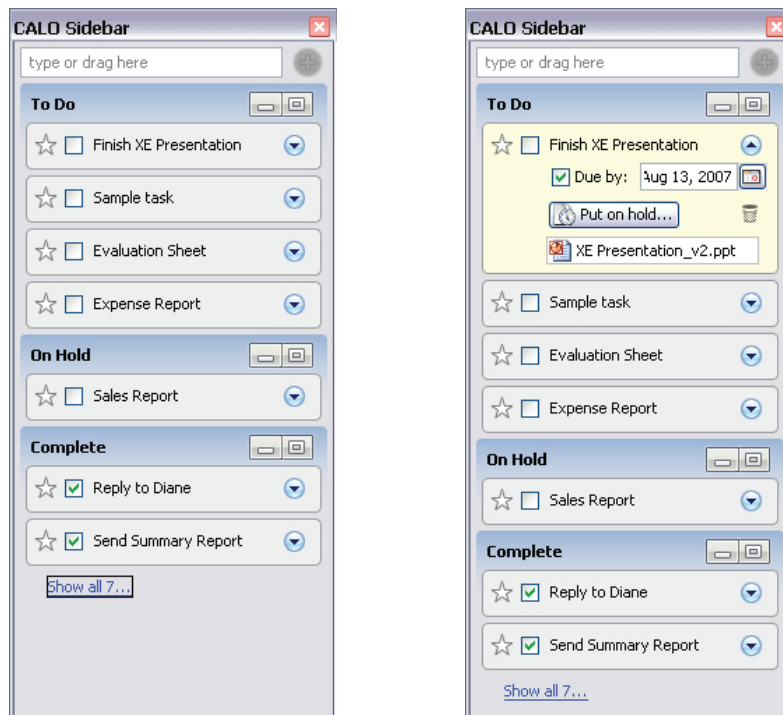
b. Think-alouds with wizard of Oz prototype

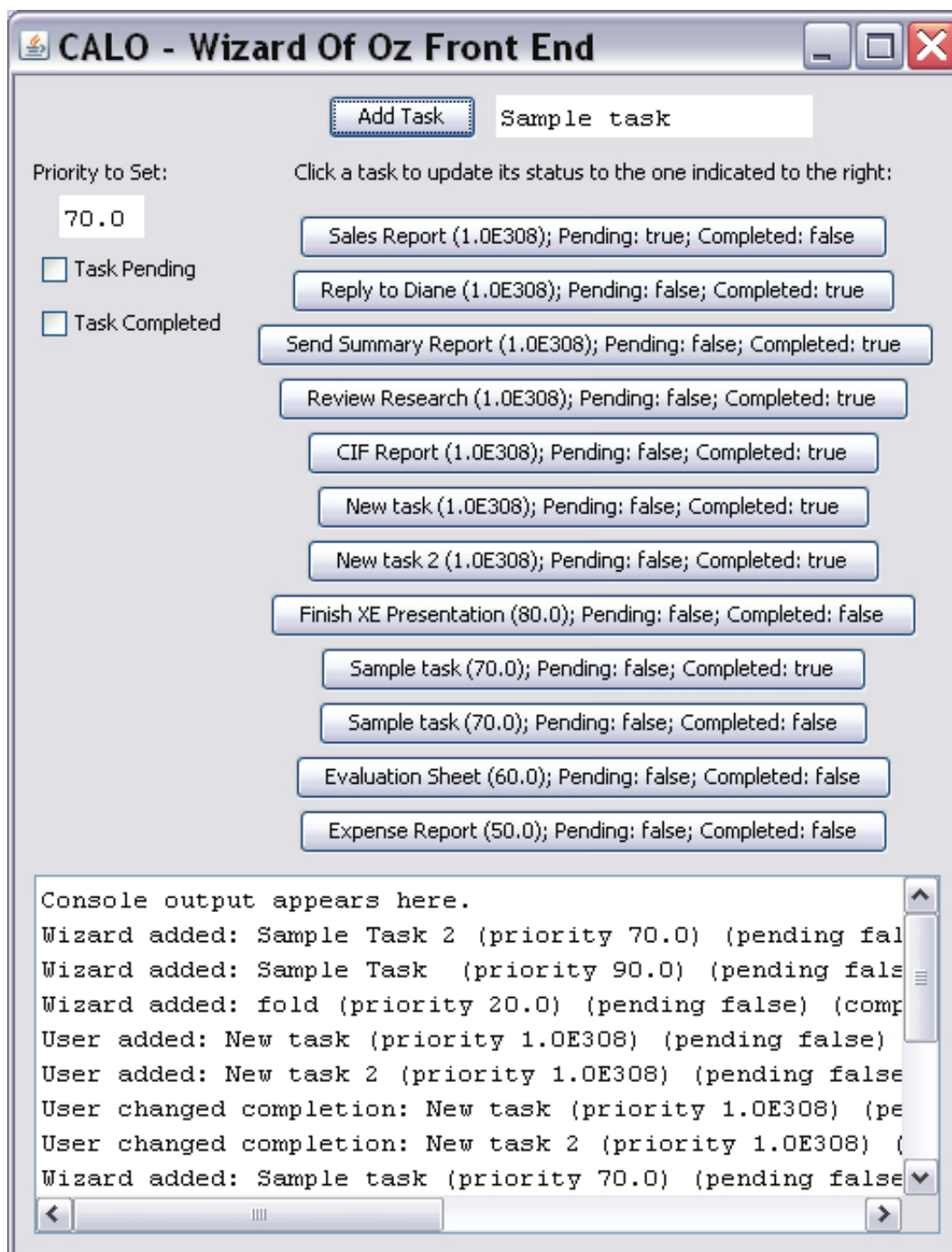
We took our findings from the paper prototypes and created an interactive prototype that employed the method of Wizard-of-Oz to simulate the AI. Users interacted with the sidebar (the task pane only, for this round of testing) on one computer, while another person used a wizard on another computer to simulate the system actions, such as adding tasks, changing priorities, and marking tasks as completed or pending, and so on. We were particularly interested in this round of testing to see how users would interact with a higher fidelity prototype where the system would automatically shift tasks around to assist them. We wanted to know if the movement was noticeable, helpful, or confusing for the system to intelligently figure out and react accordingly the user's actions. We gave users a background story to give them a sense of their work and a series of tasks that required them to interact with the task pane and an email inbox.

Our next round of user tests with an improved interactive prototype was with our target users. We felt that there was a big difference in the data collected from target users, because they understood a lot of the concepts behind our sidebar. They knew what pending tasks were without much difficulty, and they understood the reason for tasks to move between subpanes. The idea of task priority also came to them more easily than it did for others, and they appreciated some of the automatic CALO actions because of the sheer amount of work they have to handle everyday.

Think-aloud Study III: User side bar

Figure 33





Think-Aloud Study III Results

User tests with First Implementation

Users:

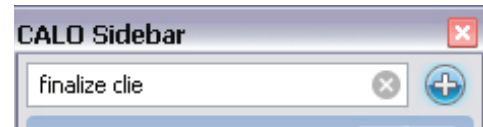
	Years of Computer Use	Computer Usage	Use digital calendar or task manager?
User 1	11	Surf the web, check email, use AIM to talk to friends, research projects and assignments, share photos etc.	Google Calender, post-it notes on my desk, and hand calender that I keep in my bag.
User 2	~12 years	Email, Internet, school, work, entertainment, scheduling events, news, etc.	Use paper planner and a calendar on my email
User 3	~ 10 years	Email, database entry, web surfing, paying bills	Use a paper schedule book
User 4	Over 20 years	Homework, e-mail, work, news, entertainment, etc.	PDA and an old fashioned calendar

For this Think-aloud, we employed the method of Wizard-of-Oz to simulate the AI. The users were non-target users that interacted with a sidebar (the task pane only) on one computer. On another computer, a person simulated the AI by adding tasks, changing priorities, and marking tasks as completed or pending, etc on the sidebar the user interacted with.

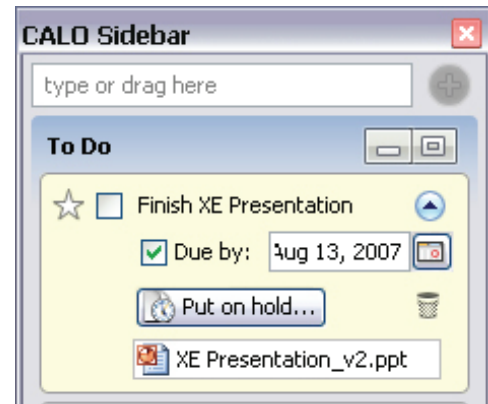
We were interested in seeing how users interacted with a higher fidelity prototype on a computer where the system automatically moved tasks around for them. We wanted to know if the movement was noticeable, helpful or confusing for the system to intelligently figure out and react accordingly to what the user is doing on the computer. We gave users a background story to give them a sense of their work and a series of tasks that seem appropriate for the position using simplified files and an email inbox.

Tasks Users Completed Without Major Problems:

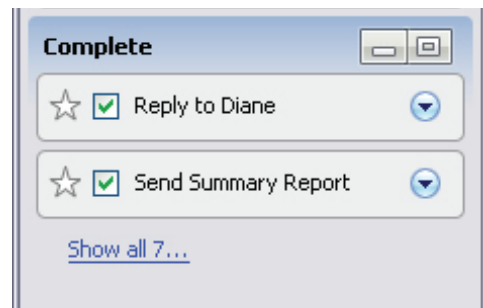
1. Adding a new task from a task field
 - Either hitting 'enter' or clicking on the add icon
2. Understand the tasks are ordered by priority
 - Most seem hesitant but tend to think it was ordered by importance or due dates which is similar to priority
3. Be able to expand tasks to see details
4. Understand the concept of completed tasks



Adding a new task



Expanded task to show details



Completed sub-pane always shows 2 tasks in the normalized mode. Maximizing or clicking on “show all 7” will show all completed tasks. Minimize button will show no completed tasks.

Problematic Areas & Tentative Solutions:

1. Noticing the order of the task added by the system

2. Noticing the system adding new tasks

- Some did, others didn't
- No one figured out where they were coming from (answer: when an email was read, the task was added) but two said it is a cool idea after we told them.
- Solution: differentiate Calo-added tasks vs. user-added tasks so that users can easily glance which tasks were added by the system and quickly check to make sure they are correct.

3. Noticing the system move the priority up for task you start working on (when they are not on the top)

- Solution: Add numbers next to task to try to encourage the idea of priority, or explain why the system is moving things to the top.

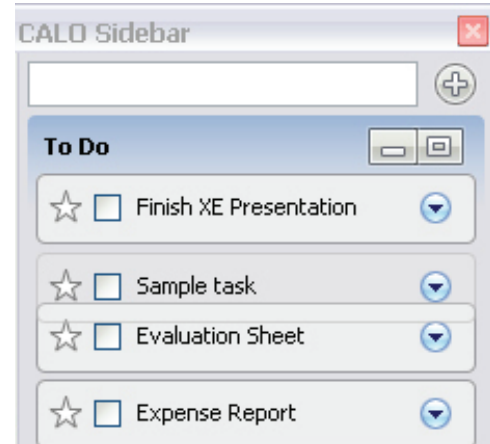
4. Be able to detect an added task

- Half of the users thought that the sender of the email added tasks for the user

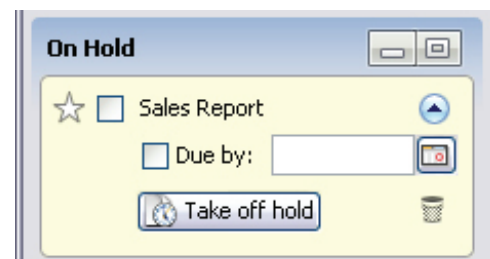
5. Understanding the concept of pending tasks

- One thought of them as all the tasks except the one they are working on
- Some thought they are tasks they decided not to work on at present (regardless of whether they are waiting on someone else)
- Solution: Add an option of leaving a note for who it's waiting on; Calo can take it out automatically if it detects pending task is no longer pending when the response comes in

6. Noticing that pending tasks automatically move to to-do after reading an email



Animation of a task changing priorities.

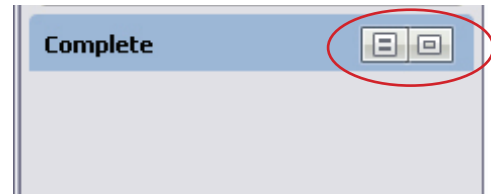


Details of Sales Report task expanded in the on-hold sub-pane. (Pending task)

- No one noticed because they were reading an email

7. Noticing the system marking tasks complete automatically

- No one noticed
- One thought it was useful
- One wanted to “cross out” (aka marking it complete) the completed tasks herself for satisfaction and keeping track of what was done and needs to be done (rather than the system marking them complete)



The completed tasks are minimized now. The icons to normalize and maximize were confusing to users

8. Be able to expand a minimized task category

- Some were ok, some were not

Solution: Have one button in the blade title that collapses or expands the pane category and keep the link to show more or less at the bottom of the list (still testing)



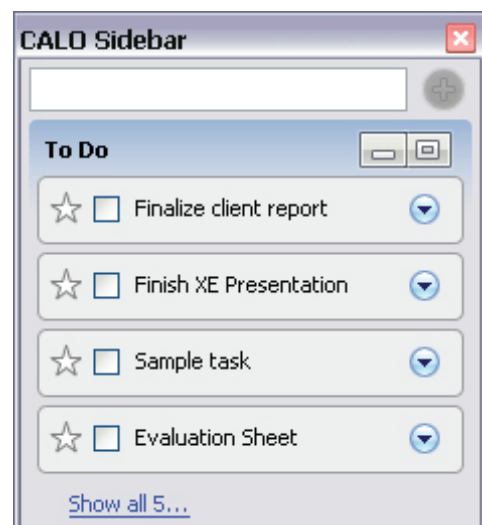
9. Being able to expand the collapsed sub-pane (to-do, on hold or complete) to see the tasks that are not visible

- Understands the interaction, but doesn't always think to click when looking for something.

10. Being able to see the new tasks added that go below the fold of visibility (less priority)

- There is very minimum feedback currently; tasks are added but not visible; the number in “show all Ns” is changed

Solution: add an animation that show the task added but disappearing below the line of visibility



When a task is added with a low priority it drops below the 4 visible tasks

Open-ended Comments from Users:

1. Was it confusing/disorienting when the system did things on its own?

- Three said yes, two said no (comfortable with similar agents ex. email spam filter)

2. Was it confusing when tasks switched their position on the sidebar?

- Though few noticed when things moved in the task pane, most thought it would be confusing

3. Would you like more control over system actions?

- Most said yes or more information of what the system did
- Some were ok as long as they also could change things

4. Would you like to be notified whenever the system makes a change?

- Most said yes
- One said that would be too much
- It's hard to say without knowing the frequency of the system making a change (we think)

5. There is no way to differentiate urgency among priority (how much more urgent than second on the list to the end?)

- Solution: Add a due date next to title of task when collapsed to see urgency at a glance

Other Issues that have come up and Changes We Made after TA III:

Concern over the lack of feedback / control when the system did something

- Even when we explained the notion of notification pane category (calo actions) some users thought that un-doing the system's action in notification center is slow and inefficient

Solution: Differentiate Calo-added tasks vs. user-added tasks so that users can easily glance which tasks were added by the system and quickly check to make sure they are correct

How we are implementing: by marking the tasks added by the system with an orange C icon which can be dimmed when users want to approve them

The combination filter/add field

- No user noticed that text field was also filtering
- Some got confused (partly because of code bug) that tasks seemed to have disappeared after accidentally filtering or leaving the text

Solution: Abandon the combination of filter/add field (how? not sure yet.)

Think-aloud IV results

Tasks users completed without major problems

1. Adding a task

Users were able to find the text field and use the enter key or click the plus button.

2. How to use the notification center

Users understood how to check and interact with the messages in the notification center

3. Concept of grouping

Some users were able to easily group items together through drag and drop. Other users did not realize you could drag and drop and therefore were not able to group them.

4. Understand concept of pending

Users understand the concept of pending, but some expanded the concept to put tasks that are unconfirmed in the “on hold” section.

5. Seeing/understanding tasks change priority

Most users thought that tasks were ordered by urgency which is a subset of priority.

Problematic areas and tentative solutions

1. Not able to notice notification

Notifications fade in and pulse for a few seconds, but many users did not notice the slight movement.

Solution: In order to capture the user’s attention, we decided to make urgent notifications stay glowing yellow.

2. CALO added a task

Some users did not realize CALO added a task if the task fell below the fold. Tasks fall below the fold when the subpane is normalized (shows only 4 tasks) and the priority of the task is lower than the top four tasks.

Solution: Show the number of total tasks and how many are currently showing, ex. (4/6) tasks

3. Marking a task as pending

Some users were unsure of what the pending icon meant.

Solution: There were too many things on the pending icon. Simplify it by taking off the arrow and animation

4. Understand how to change priority

Some users did not know how to change priority of a task especially if they didn’t realize you could drag and drop in the interface.

User comments

1. I like having this sidebar, better than maximizing an application, try to read it, it takes me so long

2. In the schedule pane, I definitely want a way to call up a day, and to see THAT DAY and see what’s due

3. Wants CALO to read her emails, see meeting requests, see what scheduling conflicts are there, and automatically respond to her emails with suggestions to what works if there are conflicts

c. Heuristic evaluation

Heuristic evaluation is a method to critically examine an interface to make sure it meets standardized usability principles such as consistency, flexibility and visibility. During heuristic evaluation, we found several areas of improvements and came up with solutions to address them.

1. Pending icon difficult to understand

Fix: To clarify the pending or “on hold” icon, we took off the red arrow and animation that occurred when the user moused over the button.

2. Auto selection of radio button for “due date” when calendar date is selected

Fix: If the user clicks on a date, the “due date” radio button becomes automatically selected.

3. Search bar does not search through tasks in subpanes that are collapsed

Fix: When the user does a search, the subpanes automatically open to show matching results even in the closed panes.

4. Subpanes may appear to be empty

Fix: Show the number of tasks visible out of the total number of tasks. For example, (4/10) means 4 tasks showing out of 10.

5. Visual distinction between panes

Fix: Create a coloring theme that makes pane division clearer.

6. Noticing incoming important notifications

Fix: Important notifications will remain highlighted in yellow until the user acknowledges it.

No. CALO-HE-01	Problem
Name: The pending icon is hard to understand and see clearly	
Evidence	
Heuristic: User control and freedom	
Interface aspect: The mouse arrow is so big that it's hard to tell what the pending icon is changed to in its hovered state. The arrow is too small, and not entirely clear.	
Severity Rating: 2	

No. CALO-HE-02	Problem
Name: Auto selection of radio button for “due date” when calendar date is selected	
Evidence	
Heuristic: User control and freedom	
Interface aspect: Making the user click on the radio button then select a due date proves too cumbersome.	
Severity Rating: 3	

No. CALO-HE-03	Problem
Name: Up & down arrows next to the month and year fields on the calendar on the task pane	
Evidence	
Heuristic: Consistency and standards	
Interface aspect: The up & down arrows seem awkward next to the down arrow for the drop down menu for month selection. Perhaps a link on the bottom of the calendar would be better? A link that says “Next month”... right now the up and down arrows do not enough affordance as to what they are going up and down through.	
Severity Rating: 2	

No. CALO-HE-04	Problem
Name: Search bar does not search through tasks in subpanes that are collapsed	
Evidence	
Heuristic: Flexibility and efficiency of use	
Interface aspect: Since the search bar is on top of all subpanes, it is the only way to search for tasks in all the subpanes. The user may have collapsed a pane just to see everything else better, and it may not make sense for the search to be limited only to expanded subpanes.	
Severity Rating: 3	

No. CALO-HE-05	Problem
Name: Subpanes may appear to be empty	
Evidence	
Heuristic: Visibility of system status	
Interface aspect: When subpanes are minimized, the user may confuse that fact with the fact that there are NO tasks under those subpanes. Subpanes should have a number on the title bar, indicating how many items are in that subpane, irregardless of what state it is in.	
Severity Rating: 3	

No. CALO-HE-06	Problem
Name: Drag and drop is a bit difficult	
Evidence	
Heuristic: Flexibility and efficiency of use	
Interface aspect: It would be nice to have a bit more space in between 2 tasks whenever another task is dragged in between them. Right now it's slightly difficult to distinguish between the in between space that the dragged item will go to, or the gray border around a task.	
Severity Rating: 2	

No. CALO-HE-07	Problem
Name: Knurling not consistent	
Evidence	
Heuristic: Flexibility and efficiency of use	
Interface aspect: Apparently, the 4 arrow grabbing cursor appears on the top of the task as well as on the bottom, but there is no knurling on the top when there is knurling on the bottom. Perhaps the grabbing cursor is sufficient? What is the purpose of the knurling if you can grab the task somewhere else without the knurling?	
Severity Rating: 2	

No. CALO-HE-08	Problem
Name: Visual distinction between panes	
Evidence	
Heuristic:	
Interface aspect: Sometimes it is hard to tell when one pane starts and ends because of the similar coloring. There are divider lines, but the dividers are not very visible.	
Severity Rating: 1	

No. CALO-HE-09	Problem
Name: Noticing incoming important notifications	
Evidence	
Heuristic: Visibility of system status	
Interface aspect: When an important notification comes in, it pulses 5 times, but is still easy to miss.	
Severity Rating: 2	

No. CALO-HE-10	Problem
Name: Pending dialogue text	
Evidence	
Heuristic: Match between system and real world	
Interface aspect: The dialogue box that comes up after clicking on pending task icon is more complicated than it should be.	
Severity Rating: 2	

No. CALO-HE-11	Problem
Name: No shortcuts in adding a new task	
Evidence	
Heuristic: Flexibility and efficiency of use	
Interface aspect: When adding a task in the text field, users cannot specify dates or other details directly in the text field. Expert users are used to adding multiple fields of content using a comma (e.g, Google Calendar) and to them, it's tedious to have to specify details later after the task is added.	
Severity Rating: 2	

No. CALO-HE-12	Problem
Name: No clear way to undo an action	
Evidence	
Heuristic: Flexibility and efficiency of use	
Interface aspect: With anything users do on the sidebar, there is no single consistent way to undo the action.	
Severity Rating: 2	

4. Specification Sheet

	Normal Sidebar	Mini-bar
General		
	<p>The normal sidebar consists of three resizable panes (task pane, schedule pane, and CALO suggestions pane), and two unresizable panes (notification center and icon well).</p>	<ol style="list-style-type: none">1. Click on the up diagonal arrow icon to turn the sidebar into the mini-bar.2. Click the down diagonal arrow on the top to bring the full sidebar back; click the close icon to close the system.3. Single click on an icon will open that pane.4. Clicking outside pane will close pane.5. Cannot right-click on mini pane icons.6. You can only open one pane at a time.7. Click the maximize icon on the top to bring the full sidebar; click the close icon to close the system.
Notification Center Features		
General	<ol style="list-style-type: none">1. The notification center cannot be collapsed.2. The default amount of space for the notification window will be for 3 notifications. If more notifications come in, the window will grow to fit them.3. There are 3 types of notifications<ul style="list-style-type: none">- Urgent emails- CALO actions- CALO added a new task	

	Normal Sidebar	Mini-bar
	<ul style="list-style-type: none"> - CALO has taken a task off hold - CALO has finished automation, etc. - Reminders <p>4. Emails and calo action notifications both group by type when they are under a certain priority. When there is an urgent email or calo action it will break from its group and have an important badge placed on the icon.</p> <p>5. Notifications in the Reminders category never group since they are usually time sensitive.</p> <p>6. Grouped notifications have email “badges” which are small icons that show the number of emails in the group. There are also important badges that for any reminders or notification that have a high enough priority to break out of a group.</p> <p>7. Clicking on the “Notification Center” title bar does nothing.</p>	
When a notification comes in	<ol style="list-style-type: none"> 1. If the notification is under a certain priority number, then the notification fades in. 2. If the notification is over a certain priority number, then it pulses yellow a couple times and stays lit. When attended to (ie. clicked on), it changes to its proper color. 3. The high-priority notification always stays on the top, even 	<ol style="list-style-type: none"> 1. When a notification comes in, a Growl-style pop-up would slide out of the mini-bar and remains yellow for two seconds while the icon on the mini-bar pulses. After that, the Growl-style pop-up slides back in and disappears, and the notification icon would stop pulsing but remain yellow until the user clicks on it. 2. If the user clicks on the Growl pop-up while it is still in view, then

	Normal Sidebar	Mini-bar
	<p>if it's not the newest. Any notification would take itself off the notification center based on an AI algorithm that would know when it's no longer necessary. For example, when the start time of a meeting has passed and there's no user activity on the desktop, then the notification for the meeting would disappear. Or an email notification would disappear once the email is read.</p>	<p>it acts just like a normal-mode notification, including the ability to dismiss.</p> <p>3. A high-priority notification always stays on the top, even if it's not the newest. Any notification would take itself off the notification center based on an AI algorithm that would know when it's no longer necessary. For example, when the start time of a meeting has passed and there's no user activity on the desktop, then the notification for the meeting would disappear. Or an email notification would disappear once the email is read.</p>
Interaction with Notifications	<ol style="list-style-type: none"> 1. Hovering over a notification: A close icon appears on the right side of the notification (for both individual and group notifications). 2. Single-click over a notification: <ul style="list-style-type: none"> - If it's a single notification (meaning not coalesced), nothing happens - If it is a coalesced notification, then it would expand the notification, showing each individual notification (ex: hover over "6 new emails" would show the subject line of each email) - If it is a highlighted notification, the highlight will turn off 3. Double-clicking on a notification will bring you to an application associated with your notification (ex: email client, or task viewer, etc.). 	<ol style="list-style-type: none"> 1. Hovering over a notification icon: Tooltips appear to show the name of the notification. 2. Single-click on a notification icon: only that notification would open up (ex: if one clicks on the "C" icon, only the CALO Action shows, and if it's a coalesced group, then all the notifications in the group would show). 3. Double-click on a notification icon: nothing happens. Same as single-click. 4. Right-click on a notification icon: nothing. 5. When a notification is opened, right-click on it would bring up a menu with "Open," "Delete," and then a "Training CALO" category with "This is incorrect," "This is almost correct," and "More Training..."

Normal Sidebar	Mini-bar
<p>4. Right-click on a notification: brings up a menu with:</p> <ul style="list-style-type: none"> - Open - Delete - “Training CALO” category with the following: <ul style="list-style-type: none"> a. This is incorrect b. This is almost correct c. More Training... 	<p>6. When a notification is opened, double-clicking on it will bring you to an application associated with your notification (ex: email client, or task viewer, etc.).</p>

Task pane components

General	<p>1. Each subpane has 3 states: collapsed, normal, and expanded.</p> <p>2. Click the triangle inside a circle to collapse each entire subpane. The tasks in a collapsed subpane are now hidden.</p> <p>3. The normalized state would show to-do tasks, on hold tasks and completed tasks in a ratio of 4:2:2, multiplied out to fit the entire height of the pane. Right now, it shows 4:2:2 regardless of height.</p> <p>4. The expanded state is set by clicking on the “show all n” link at the bottom of each subpane to show all tasks in the subpane. Each group’s expanded state would have a link below the items that says “Show top #” where “#” is the normalized number described above in 3.</p> <p>5. When the task pane has more items than space available, a scrollbar appears.</p> <p>6. When a task is added by CALO, a C icon is included in the task. Clicking on the C icon makes it grayed out. Clicking on it again will make it colored again.</p>	<p>1. Single-click on the task icon: opens the task pane.</p> <p>2. Double-click on the task icon: opens the task pane (same as single click).</p> <p>3. Once the task pane is opened, all other functions are the same as the task pane in the normal sidebar.</p> <p>4. Clicking outside of the mini sidebar would close any opened panes.</p>
---------	--	--

	Normal Sidebar	Mini-bar
	7. When a due date for a task is added, the due date is visible even when the task is collapsed.	
Title task bar	1. Single-click on the expand icon would collapse/expand the whole pane. 2. Double-click would open the task viewer.	
Text field	1. Typing in the add text field will add a new task by pressing enter or clicking on the “+” 2. Typing in the search field will filter the task list. To clear the filter, delete the text or click on the x button. 3. You can drag documents onto the add text field to add a task or drag the document onto the task pane. 4. Dragging a document onto a particular task will add it as a resource to the task.	
To Do subpane	1. The normal state would have 4 task items. 2. Single-click on the star would toggle between a star outline and a solid star. (refer to Gmail or Towel) 3. Single-click on the Completion button makes the arrow outline solid, but once the button is clicked, the task would be moved to the “Completed” group. 4. Single-click on the down arrow	

Normal Sidebar

Mini-bar

of a task would expand the task, showing its details. Clicking on the up arrow will collapse the task.

5. Double-click on a task would bring out the task viewer with that particular task highlighted and expanded.

6. Right-click on a task would bring up a menu with “Delete,” “Star,” “Mark as complete,” “Put on hold,” and a “Training CALO” category with “This is incorrect,” “This is almost correct,” and “More Training...”

7. Clicking on the trash can icon would delete the task.

8. An expanded task would have the following details:

- Due by field
- Tags
- Put on hold button (moves the task to on hold)
- Resources related to the task which include files, emails, etc.
- Add a resource icon.
- Make a pack icon (not shown or implemented)

On Hold subpane

1. The normal state holds 2 task items by default.

2. Right click on a ‘on hold’ task would bring up

- Delete
- Star
- Mark as complete
- Take off hold
- Training CALO category with
 - a. This is incorrect
 - b. This is almost correct
 - c. More Training...

	Normal Sidebar	Mini-bar
Complete subpane	<ol style="list-style-type: none"> 1. The normal state has two task items. 2 Right clicking on a task in the completed group would bring up <ul style="list-style-type: none"> - Delete - Star - Mark as incomplete - Put on hold - Training CALO” category with: <ol style="list-style-type: none"> a. This is incorrect b. This is almost correct c. More Training... 3. Items in the completed subpane are already checked. Unchecking the task will push it back to To Do. 	

Task viewer		
General	<ol style="list-style-type: none"> 1. Double clicking on a task in the side bar will bring up the a separate window called the Task Viewer with that particular task expanded to see the details. 2. Drag and drop doesn’t work to move tasks around, but it does work to create groups. 3. Pending and completed are mixed in with completed tasks, not separated in groups. <ul style="list-style-type: none"> - Completed tasks are identified by the check mark button filled in. - Pending tasks have a pending status icon 4. Tasks are by default listed by priority. 5. Grouped tasks looked the same as in task pane. 	<ol style="list-style-type: none"> 1. Clicking on the icon with the document and magnifying glass will open the task viewer in a new window.

	Normal Sidebar	Mini-bar
Task viewer interaction	<ol style="list-style-type: none"> 1. The left text input box creates new task. 2. The right text input box allows you to search through tasks. 3. The row of 'buttons' across the top are filters: deadline, date added, project, tags, status and people <ul style="list-style-type: none"> - Hovering over a filter button will make a down arrow appear. - Clicking on the text of the filter button will sort the tasks by the filter name - Clicking on the down arrow opens a menu with options to filter the list on Ex. people will have a list of people, clicking on one will only show tasks related to that person - When a menu button is filtering on a dimension, there is a check by the category name to indicate what categories are being sorted. 4. Clicking on a task expands it to see its details. There are more details listed here than in the task pane. <ul style="list-style-type: none"> - Title, date added, due date, tags, people, etc. - More resources are listed 5. Clicking on any of the details will allow the user to edit them ex. tags or date due. 	<ol style="list-style-type: none"> 1. Clicking on the icon with the document and magnifying glass will open the task viewer in a new window.
Contextual menu	<ol style="list-style-type: none"> 1. Right click on the task gives you the same as the task pane which varies depending on whether it is a incomplete, on hold or complete task. Refer to task pane. 	

2. Right click on a resource/
document brings up menu with
 - Remove document
 - Find similar files
 - This is incorrect
 - This is almost correct
 - More training

Schedule components

- | | |
|------------|---|
| General | <p>Schedule pane has two views:
block view and agenda view</p> <ul style="list-style-type: none"> - Users switch the view by either: <ul style="list-style-type: none"> - Clicking on the agenda and the block icon at the bottom or - Zooming in and out. <p>When in the block view zooming all the way out brings the user to the agenda view.</p> |
| Block view | <ol style="list-style-type: none"> 1. View: <ul style="list-style-type: none"> - The hours are represented by horizontal lines 2. Anchoring: <ul style="list-style-type: none"> - The schedule is automatically anchored to one 30 minutes before the present time (this is why there is no scroll bar). 3. Zooming: <ul style="list-style-type: none"> - The default view is showing 5 hours in the window. Resizing the schedule pane will allow it to show more or less at one time - Zoom out to view more of the day - Zoom in to view fewer hours - When users zooms out far (toward -), the view switches to agenda view |

4. Events:
 - Events are displayed as blocks of time
 - Users can drag the bottom of the block to change the duration of events (not implemented)
 - Pack icon is visible when events have associated resources
5. Single click:
 - Single clicking on events would open the details of the event such as starting and ending time, location, people and notes. It would also show a trash can.
 - Users can exit the detail view by clicking anywhere outside the block.
 - Users can click on the texts to modify information
6. Adding a new event:
 - Users can add a new event by clicking on the “add” button at the bottom and selecting a time of the day or
 - Clicking on any open area in the schedule would create a new block (1 hour default); then the user can modify length or title of the event

**Agenda
view**

- View:**
1. It shows 5 days of schedule in a list.
 2. When the list cannot fit in the pane, a scroll bar appears.
 3. The list contains the starting

Normal Sidebar	Mini-bar
time, title of each event and a Pack icon when applicable.	
<p>Anchoring:</p> <ol style="list-style-type: none"> 1. It is anchored to the current day. 	
<p>Zooming:</p> <ol style="list-style-type: none"> 1. Zooming in (toward +) would switch the view to block view. 	
<p>Events:</p> <ol style="list-style-type: none"> 1. Events are editable by clicking on the texts. 	
<p>Adding a new event:</p> <ol style="list-style-type: none"> 1. Users can add an event in next 5 days. 	

CALO suggestions components

General	<ol style="list-style-type: none"> 1. The CALO suggestion pane will show files or emails related to the form and actions the system can perform. In this case, the CALO suggestion pane gives contextual or relevant information and resources related to the window that the user is focused on. For example, if the user has a purchase form in focus the CALO would be able to automate this task by filling in the fields for the user. 2. When a pane in the side bar is in focus, CALO suggests will give descriptions and directions on the actions available in that pane. 	<ol style="list-style-type: none"> 1. Every time calo suggests has an available action, the icon will be colored.
---------	--	--

	Normal Sidebar	Mini-bar
Contextual menus	<p>Right clicking on a document will bring up a contextual menu:</p> <ul style="list-style-type: none"> - Remove document - Find similar files - Training CALO” category <p>with:</p> <ul style="list-style-type: none"> a. This is incorrect b. This is almost correct c. More Training... 	

Icon well components

- General
1. Task Viewer icon:
Opens the task viewer window
 2. Calendar icon:
Opens the user’s own calendar application
 3. Contacts icon:
Lists contacts the system has collected for the user
 4. Email icon:
Opens the user’s email client
 5. Automation icon:
Opens automation window for system to complete a repetitive task for a user.
 6. Learning log icon:
One section lists all important user actions which are documented here. Another sections lists all actions Calo did with the ability to train or correct based on any past action Calo took.
 7. Preferences icon:
Allows the user to change settings of sidebar
 8. Help icon:
Opens the help window

Pack

General

A pack is created to automatically gather all resources into one folder. Packs can be made for meetings, tasks and eventually entire days. A window explorer is on the left hand side for easy access to documents.

1. View:

Allows the user to select icon, list, or thumbnail view.

2. Actions

The action buttons are along the top

- Add
- Remove or remove selected
- Print
- Save

3. Filtering

Filtering works the same was as the task viewer. Refer to task viewer.

4. Search

A search bar is provided to enter keywords.

5. Adding files

Files can be added using the Add action button or by dragging files into the pack.

6. Removing files

Files can be removed by dragging them out of the pack window.

Right click on a file brings up a contextual menu with

- Remove document
- Find similar files
 - This is incorrect
 - This is almost correct
 - More training...

VI. Bibliography

1. Adamczyk, P.D., Bailey, B.P., A Method and System for Intelligent Interruption Management. in Task Models and Diagrams, (2005).
2. Avrahami, D., Hudson, S. E., QnA: Augmenting an Instant Messaging Client to Balance User Responsiveness and Performance. in ACM Conference on Computer Supported Cooperative Work, (2004).
3. Bao, X., Herlocker, J., Dietterich, T., Fewer Clicks and Less Frustration: Reducing the Cost of Reaching the Right Folder. in IUI, (Sydney Australia, 2006).
4. Barthelmess, P., Kaiser, E., Lunsford, R., McGee, D., Cohen, P., Oviatt, S Human-Centered Collaborative Interaction.
5. Bellotti, V., Ducheneaut, N., Howard, M., Smith, I. Taking Email to Task: The Design and Evaluation of a Task Management Centered Email Tool.
6. Berry, P., Myers, K., Uribe, T., and Yorke-Smith, N., Task Management under Change and Uncertainty: Constraint Solving Experience with the CALO Project. in CP 2005 Workshop on Constraint Solving under Change and Uncertainty, (Sitges, Spain, 2005).
7. Berry, P., Peintner, B., Yorke-Smith, N. , Bringing the User back into Scheduling: Two Case Studies of Interaction with Intelligent Scheduling Assistants. in AAAI Spring Symposium on Interaction Challenges for Artificial Assistants, (Stanford, CA, 2007).
8. Berry, P.M., Gervasio, M., Uribe, T, Pollack, M., and Moffitt, M., A Personalized Time Management Assistant. in AAAI Spring Symposium Series, (Stanford, CA, 2005).
9. Berry, P.M., Gervasio, M., Uribe, T., Myers, K., and Nitz, K., A Personalized Calendar Assistant. in AAAI Spring Symposium Series, (Stanford, CA, 2004).
10. Biehl, J.T., Bailey, B. P., Improving Interfaces for Managing Applications in Multiple-Device Environments. in AVI, (2006), 35-42.
11. Boardman, R., Sasse, M. Stuff Goes into the Computer and Doesn't Come Out. in A Cross-tool Study of Personal Information Management, 2004.
12. Cheyer, A., Park, Jack, Giuli, Richard., IRIS: Integrate. Relate. Infer. Share. in ICSC (Galway, Ireland, 2005).
13. Chklovski, T., Gil, Y., Enhancing Interaction with To-Do Lists Using Artificial Assistants. in AAAI Spring Symposium on Interaction Challenges for Artificial Assistants, (Stanford, CA, 2006).
14. Conley, K., Carpenter, J., Towel: Towards an Intelligent To-Do List. in AAAI, (2006).
15. Dabbish, L.L., Kraut, R., Fussell, S., Kiesler, S., Understanding Email Use: Predicting Action on a Message. in ACM Conference on Human Factors in Computing Systems, (Portland, Oregon, 2005), 691-700.
16. Dietterich, T., Kaelbling, L., Marx, Z., Rosenstein, M. , Transfer Learning with an Ensemble of Background Tasks. in NIPS Workshop on Transfer Learning, (Whistler, BC, 2005).
17. Ehlen, P., Niekrasz, J., Purver, M., A Meeting Browser that Learns. in AAAI, (2007).

18. Faulring, A., Myers, B. A., Enabling Rich Human-Agent Interaction for a Calendar Scheduling Agent. in Conference on Human Factors in Computing Systems Extended Abstracts, (Portland, Oregon, 2005).
19. Fogarty, J., Ko, A. J., Aung, H. H., Golden, E., Tang, K. P., Hudson, S., Examining Task Engagement in Sensor-Based Models of Human Interruptability. in ACM Conference on Human Factors in Computing Systems, (Portland, Oregon, 2005), 331-340.
20. Garera, N., Rudnicky, A. I., Towards a Personal Briefing Assistant. in AAAI Spring Symposium, (2005).
21. Grosz, B., Sidner C. Plans for Discourse. in Intentions in Communication, Bradford Books/MIT Press, Cambridge, MA, 1990, 417-444.
22. Henderson, S. Genre, Task, Topic, and Time: Facets of Personal Digital Document Management.
23. Huang, X., Oviatt, Sharon L., Toward Adaptive Information Fusion in Multimodal Systems. in MLMI, (2005), 15-27.
24. Iqbal, S.T., Bailey B. P., Understanding and Developing Models for Detecting and Differentiating Breakpoints during Interactive Tasks. in ACM Conference on Human Factors in Computing Systems, (2007).
25. Kaiser, E., SHACER: a Speech and Handwriting Recognizer. in ICMI, (2005).
26. Kaiser, E., Demirdjian, D., Gruenstein, A., Li, X., Niekrasz, J., Wesson, M. and Kumar, S., Demo: A Multimodal Learning Interface for Sketch, Speak and Point Creation of a Schedule Chart. in ICMI, (State College, PA, 2004), 329-330.
27. Kaiser, E.C., Dynamic New Vocabulary Enrollment through Handwriting and Speech in a Multimodal Scheduling Application: Making Pen-Based Interaction Intelligent and Natural. in AAAI Symposium Technical Report FS-04-06, (Arlington, VA, 2004), 85-91.
28. Kaiser, E.C., Barthelmess, P., Huang, X. Demirdjian, D. A, Demonstration of Distributed Pointing and Referencing for Multimodal Collaboration Over Sketched Diagrams. in ICMI, (Trento, Italy, 2005).
29. Kaye, R., Karam, G., Cooperating Knowledge-Based Assistants for the Office
30. Klimt, B., Yang, Y., The Enron Corpus: A New Dataset for Email Classification Research. in European Conference on Machine Learning, (2004).
31. Kozierok, R., Maes, P., A Learning Interface Agent for Scheduling Meetings. in IUI, (1993).
32. Krause, A., Siewiorek, D. P., Smailagic, A., Farrington, J., Unsupervised, Dynamic Identification of Physiological and Activity Context in Wearable Computing. in IEEE International Symposium on Wearable Computers, (New York, NY, 2003), 88-97.
33. Lerman, K., Gazen, C., Minton, S. and Knoblock, C. A., Populating the Semantic Web. in AAAI 2004 Workshop on Advances in Text Extraction and Mining, (2004).
34. Lunsford, R., Kaiser, E., Barthelmess, P., Huang, X., Managing Extrinsic Costs via Multimodal Natural Interaction Systems. in CHI, (Montréal, Québec, Canada, 2006).
35. Maes, P. Agents that Reduce Work and Information Overload. Communications of the

ACM, 37 (7). 30-40.

36. Maheswaran, R., Tambe, M., Varakantham, P., and Myers, K. Adjustable Autonomy Challenges in Personal Assistant Agents: A Position Paper. *Autonomy*, 2003.
37. Martin, I., Jose, J.M., Fetch: A Personalised Information Retrieval Tool. in *RIAO*, (2004).
38. Modi, P.J., Veloso, M., Smith, S., Oh, J., CMRADAR: A Personal Assistant Agent for Calendar Management. in *Agent Oriented Information Systems*, (2004).
39. Papazoglou, M.P. Agent-oriented Technology in Support of e-business. *Communications of the ACM*, 44 (4). 71-77.
40. Pollack, M., Weber, J., Effective Interaction Strategies for Adaptive Reminding. in *AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*, (2007).
41. Rich, C., Sidner, C., COLLAGEN: When Agents Collaborate with People. in *International Conference on Autonomous Agents*, (1997).
42. Schurr, N., Varakantham, P. Bowring, E., Tambe, M., Grosz. B. Asimovian Multiagents: Applying Laws of Robotics to Teams of Humans and Agents. *AAMAS*.
43. Shen, J., Li, L., Dietterich, T., Herlocker, J., A Hybrid Learning System for Recognizing User Tasks from Desktop Activities and Email Messages. in *International Conference on Intelligent User Interfaces*, (Sydney, Australia, 2006), 86-92.
44. Tambe, M., Bowring, E., Pearce, J.P., Varakantham, P., Scerri, P., Pynadath, D., Electric Elves: What Went Wrong and Why. in *AAAI Spring Symposium on What Went Wrong and Why*, (Stanford, CA, 2006).
45. Tomasic, A., Zimmerman, J., Simmons, I., Linking Messages and Form Requests. . in *IUI*, (2006).
46. Cassin, B. and Solomon, S. *Dictionary of Eye Terminology*. Triad Publishing Company, Gainesville, Florida, 1990.