Improving Notifications, Phase One. Actionable Notifications: Vision and Design.

Author: Edwin Li
Interaction Designer
Email: hci@edwinli.com
Phone: 510-371-4285
Website: edwinli.com/ux
LinkedIn: www.linkedin.com/in/edwinli
Last updated: April 11, 2013

CONFIDENTIAL

Contents

Overview 3
Objectives 3
Current System(s) Summary Description and Evaluation 4
webOS (Versions 1.x, 2.x, and 3.x) Notifications: Among the Best 4
iOS v6.x Notifications: Playing Catch Up 14
Android v4.2 Jelly Bean: A Quick Look 16
New System Requirements 16
Goal of Phase One 16
Inspiration: “Delete From Notifications” Patch (webOS) 17
Inspiration and Framework: Getting Things Done 18
Parallel Work 19
Process 21
Process Steps Represented In This Document 22
Artifacts Included in This Document 23
Data, Digesting the Data, and The Beginnings of Design 24
Design Directions 28
Choice of Operating System and Device 28
Improved Notifications, v0 29
Exploring More Than One Design 29
Improved Notifications v0.1 36
Considerations From v0 36

CONFIDENTIAL

Overview

Notifications—here meaning those provided on smartphones, tablets, and computers—are, by definition, intended to interrupt users' attention and are meant to inform users of things that they presumably care about as they occur.

Benefits to a useful notification system include:

- Only as obtrusive as required by the importance of the notification. This suggests more than one level of interruption, from a modal alert (one that demands that users interact with it before going on), to a notification that perhaps doesn't even call for users' attention, perhaps being only available when users seek them out. The less intrusive the notification, the greater the freedom users will have in controlling their own flow of activity in using the device or ecosystem in question.
- Persistence until dismissed by users—this allows the notification to serve as a reminder of something that users will want to do. In this way, notifications can serve as a list of things to do for users.
- Ability to go from a notification to the context within which users can interact fully with the subject of the notification, e.g., an email notification would enable users to view and react to the particular email that triggered the notification.
- User control. Ideally, users can customize which conditions, apps, or services, are allowed to provide notifications and at what level of obtrusiveness.

All designs can be improved, and the purpose of this design exercise is to propose designs that improve on the current state of notification systems (as seen primarily in webOS v3.0.x, iOS v6.1.2, with a limited view of Android OS v4.1.2), exploring first "actionable notifications," and then taking a peek beyond.

Objectives

The objective of this design exercise is to create designs that explore how notification systems in mobile OSes and beyond can be improved. The first phase in that exploration, and the phase described in this document, will be fleshing out a design for "actionable notifications", seen as part of Android OS v4.2 Jelly Bean, and presaged by other examples from which the design explored here draws its primary inspiration.

Left to further work, I believe that a larger shift in how notification systems could be designed will come from re-conceptualizing notifications as new entries in the time stream of a user's digital ecosystem, something I've taken to calling their "personal digital infrastructure" to denote individual and heterogeneous collection of hardware, software, content, and services that they use that is probably not aligned with any single mobile OS.

CONFIDENTIAL

Current System(s) Summary Description and Evaluation

The notification systems with which I have the most direct, everyday experience are those of webOS (versions 3.x and 1.x) and iOS version 6.x.

webOS (Versions 1.x, 2.x, and 3.x) Notifications: Among the Best

When webOS was announced and later released in 2009, among its most trumpeted and lauded features was its system of "unobtrusive notifications". Consistent with the primacy of multitasking in webOS' design, notifications typically showed up in a notification area at the bottom of the screen (in webOS versions 1.x and 2.x; top of the screen for webOS v3.x) in two primary heights—banner and dashboard formats. The notifications showed themselves to users but without forcing users to interact with them. If users did not interact with them, they faded away in diminished, icon form into that same notification area, persistent and visible. See Figure 1 through Figure 3 for the phone view (webOS v2.x), and Figure 5 through Figure 6 for the tablet view (HP Touchpad, webOS v3.x).

In webOS phones (webOS versions 1.x and 2.x), the app that users were using when the notifications came in would simply resize to accommodate the appearance of the notification. In the HP Touchpad (webOS v3.x), the notification area moved to the status bar at the top of the screen, removing the need for apps to resize as notifications appeared or were dismissed.

When users wanted to see their notifications in more detail than the icon form, they could expand the notifiation area to view the "dashboard", where the fuller notifications list were displayed (Figure 4 for webOS 2.x and Figure 7 for webOS 3.x).

Modal notifications were also possible, but—in contrast to iOS (the primary mobile OS of the time) in which all notifications were modal—webOS' modal notifications were intentionally infrequent. Even incoming phone call notifications were shown as "dashboard" notifications rather than modal alerts if users were actively using their phone.

CONFIDENTIAL



Figure 1 The "desktop" of webOS v2.x (from a Pre 3), showing the full size of the desktop, which has several app "cards" minimized on it. Note the absence of a black strip at the bottom of the screen, which is where the notification area appears.

CONFIDENTIAL

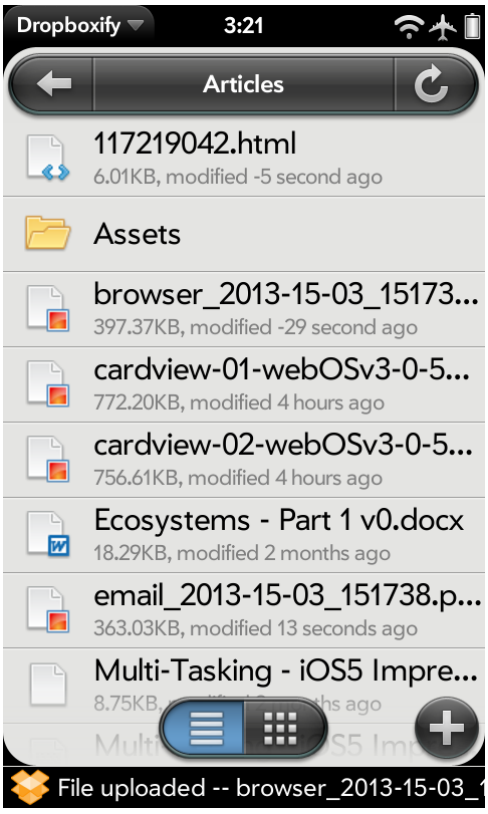


Figure 2 A banner notification from webOS v2.2.4 (from a Pre 3). True of webOS v1.x and 2.x, banner notifications consisted of the source app's icon and the notification's message, scrolling right to left once, becoming just one icon among potentially many in the notification area afterwards.

CONFIDENTIAL

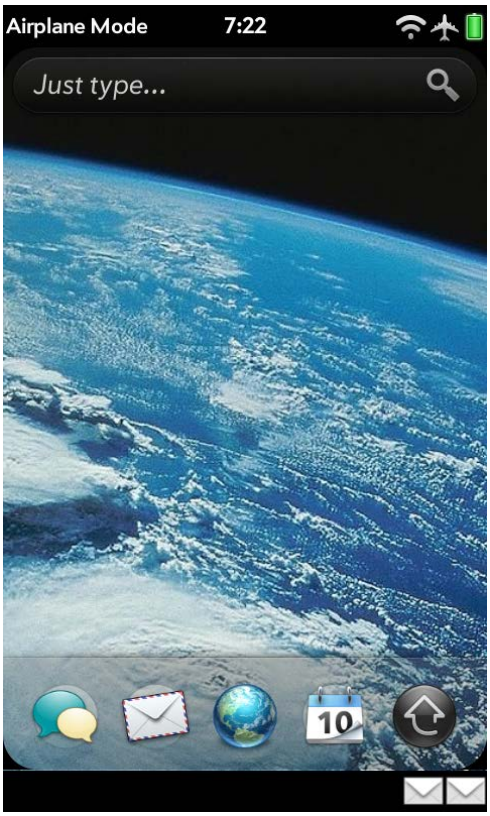


Figure 3 Notification area in webOS v2.2.4 (from a Pre3). Notifications for incoming email on two separate email accounts, waiting for the Pre 3's user to investigate and interact with them.

CONFIDENTIAL

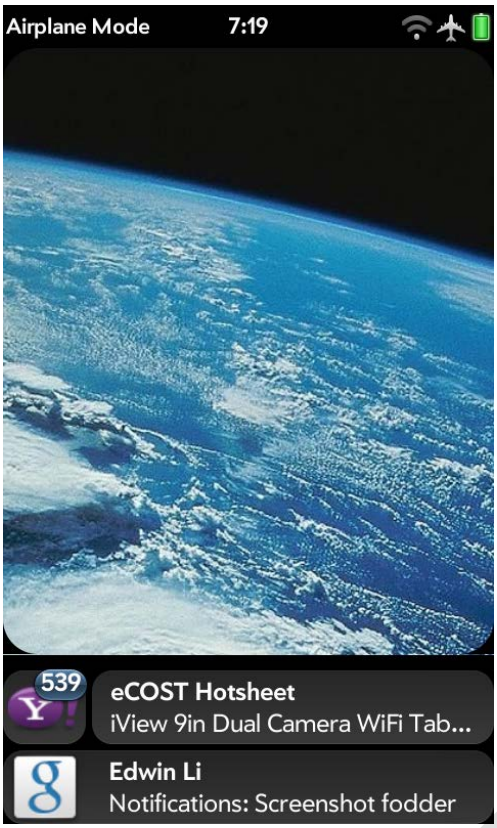


Figure 4 Dashboard view in webOS v2.2.4 (from a Pre 3). When expanded, the notification area shows a detailed view of the most recent notification in each app that has a notification pending.

CONFIDENTIAL

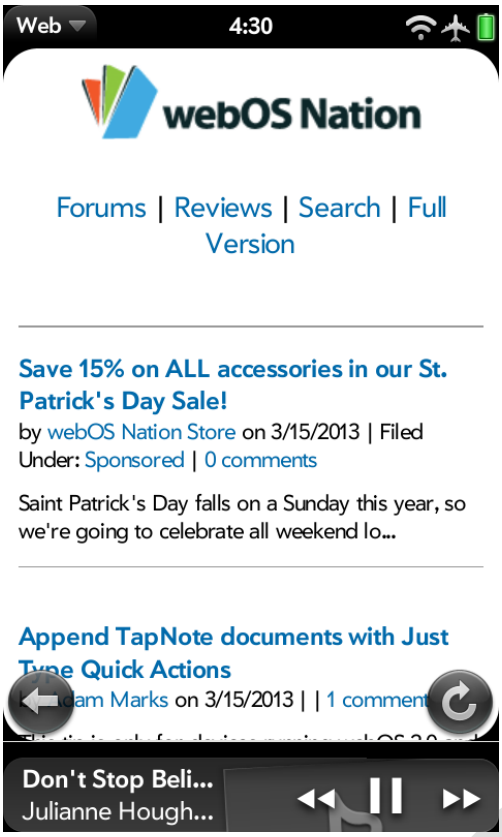


Figure 5 The dashboard view of the webOS notification area could also house dashboard controls or apps, such as this dashboard controller for the built in music player, which allows users to control music playback from anywhere in the OS rather than having to foreground the music player app. This was an aspect of webOS' multi-tasking foundation. webOS v3.x also supports this, as pictured in Figure 8.

CONFIDENTIAL

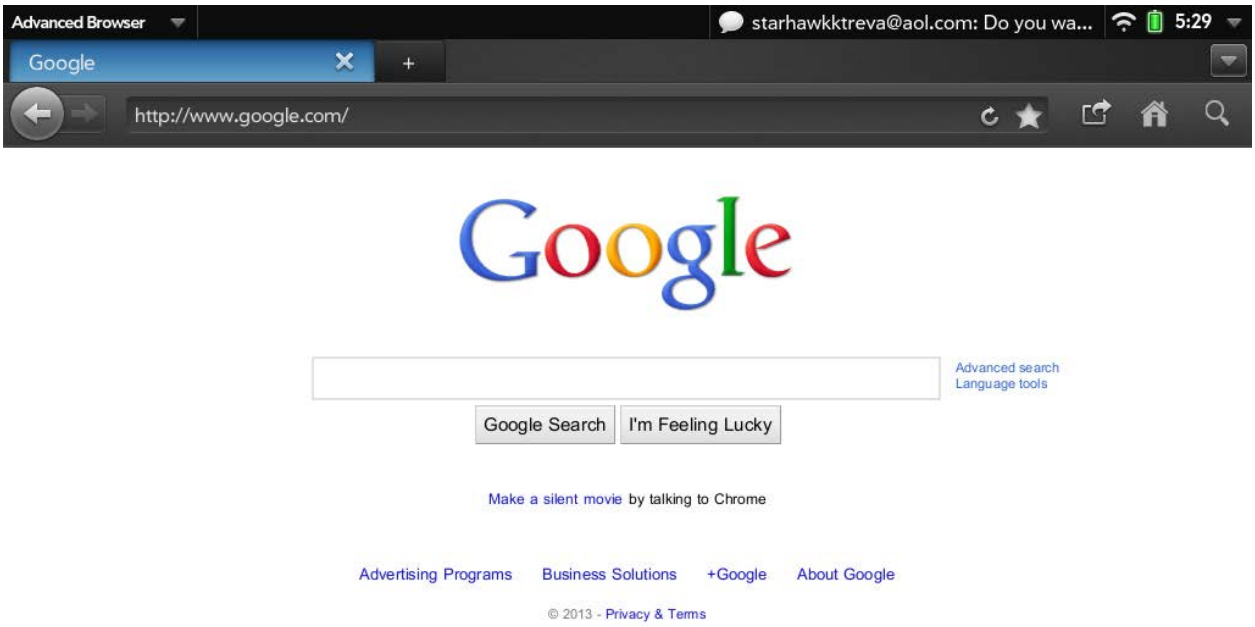


Figure 6 Incoming chat banner notification in webOS v3.0.5 (from HP Touchpad). As with webOS v1.x and 2.x, a banner notification consists of an app icon and the text content. In webOS v3.x, the notification area was moved to status bar at the top of the screen, removing the need for applications to resize as notifications come in and as users interact with them.

CONFI

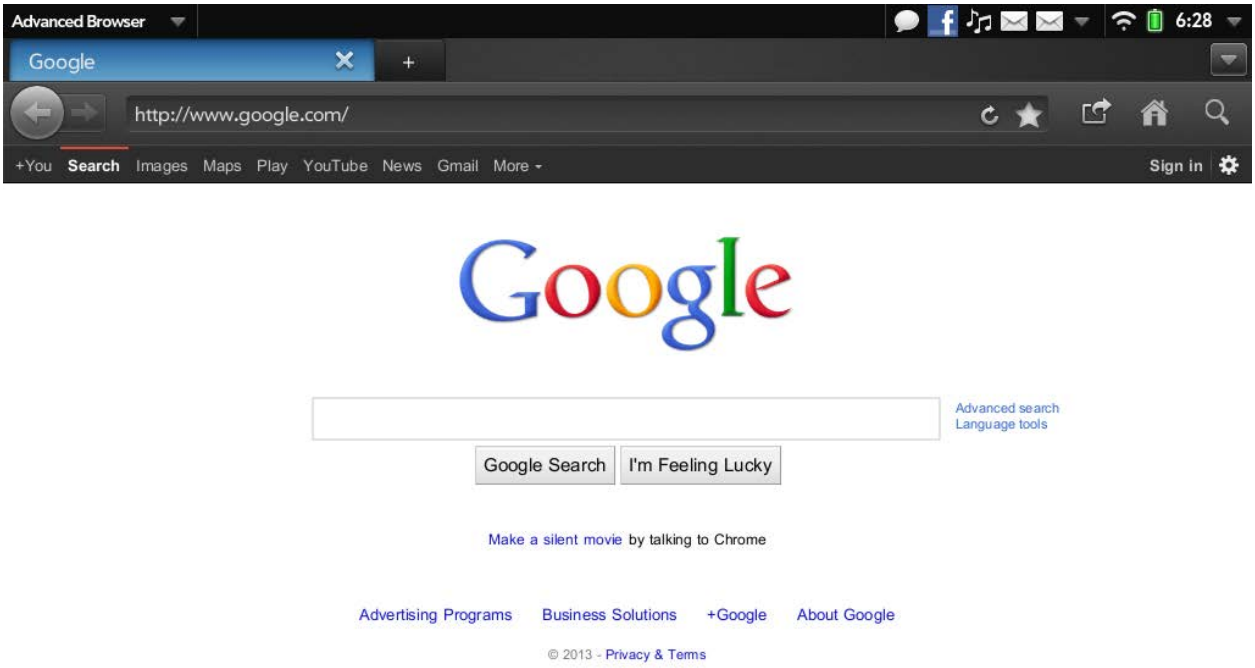


Figure 7 Notifications, minimized in webOS v3.x (HP Touchpad). As with previous webOS versions, notifications became just icons in the notification area when users choose not to interact with them.

CONFIX

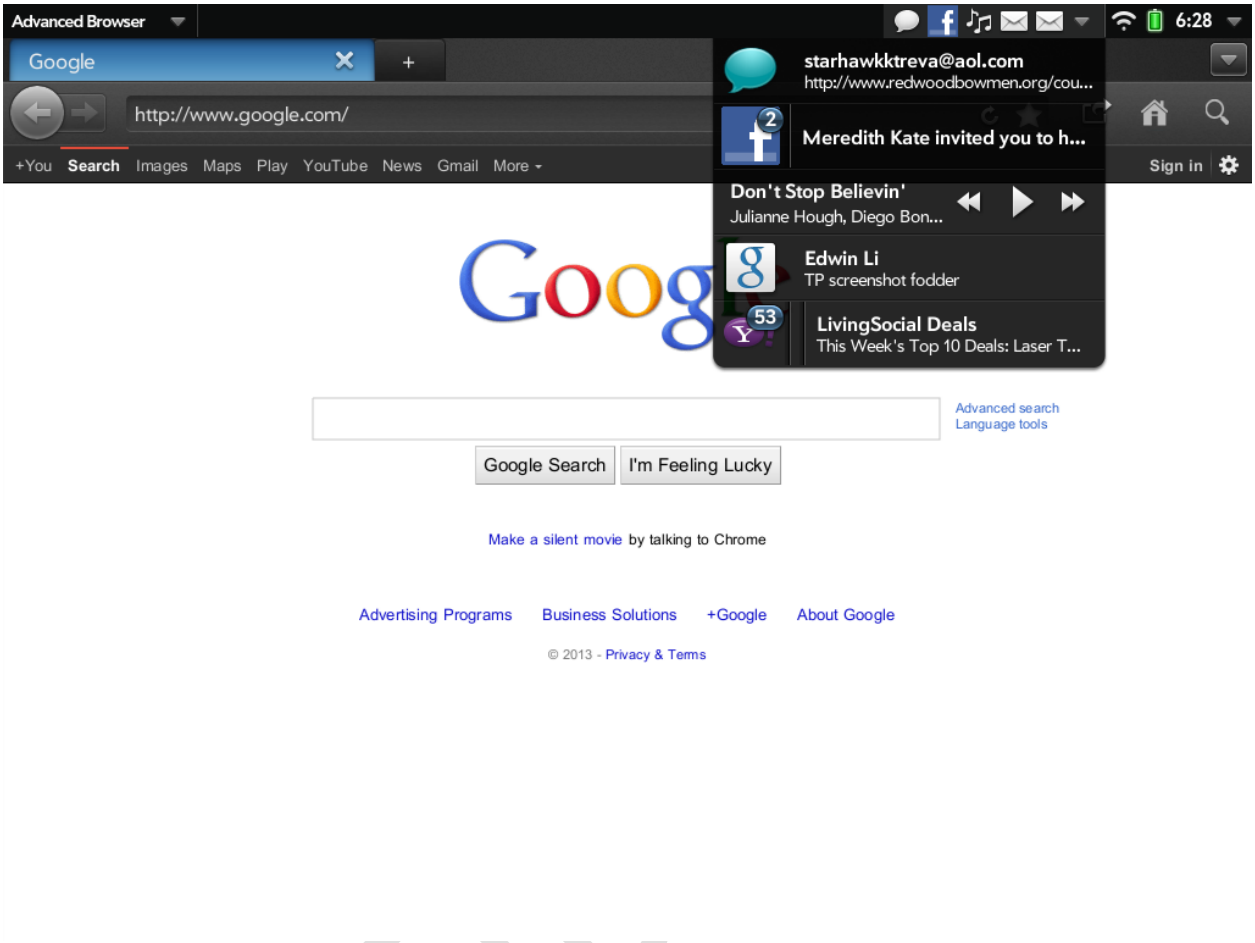


Figure 8 Notifications dashboard in webOS v3.x (HP Touchpad). When expanded, the notifications area dropped down into a notification dashboard, showing the notifications. As with webOS v2.2.4 seen in Figure 5, the dashboard here also supports dashboard controls and apps. This dashboard expands to a set maximum height, beyond which the list of notifications scrolls within that maximum frame height.

In webOS 3.x, notifications were given additional functionality. When any particular app had a more than one notification waiting for the user, those notifications would stack on top of each other. As seen with the Facebook and Yahoo Mail notifications in Figure 8, users could swipe the app icon to the left or right to dismiss all of the notifications from that app, or they could swipe to the right starting from just to the right of the vertical score line(s) to the right of the app icon to dismiss the “topmost”, visible notification, revealing the next notification in that stack. Tapping on the app icon took users to the app. Tapping on the notification’s text portion took users to that particular notification’s item in the app.

From its first announcement in January 2009, and later enhanced, webOS notifications thus provided all or most of the benefits listed in the Overview section for a good notification system:

- Unobtrusive notifications
- Persistence until user dismisses them.

    - As an added bonus here, the notifications are persistent not only in concept, but also visually throughout the OS via a visually persistent notification area, which is visible whether users are in an app or on the "desktop" of their device.
- Tapping on a notification's dashboard item takes users to the relevant app and the specific content of that notification.
- User control of which apps were allowed to trigger notifications. Unlike iOS, most of the settings for apps resided within each app. As a result, control over which apps were allowed to trigger what kinds of notification was primarily dependent on how each app was designed.

Taking webOS 3.x as the current pinnacle of the OS, webOS notifications still leave some functionality to be desired. In Android 4.2 Jelly Bean, announced October 29, 2012, at least some notifications are "actionable"—that is, they offer more functionality to users beyond either dismissing the notification or going into the app.

This is not an idea without precedent in webOS, though one would have to look into the world of webOS "homebrew" to find perhaps its first appearance.

One of the best things about webOS was its openness; developers were free to alter a great deal of how it worked and looked without barriers like "jailbreaking" from the world of iOS or "rooting" from the world of Android.

One such alteration to webOS, called a "patch", was of the email dashboard notification. That particular patch was called "Delete from Notification" (created in 11/2009 for webOS 1.x, discussed in its webOS Nation forum thread: http://forums.webosnation.com/webos-patches/215142-patch-email-delete-notification-bar.html), and it added a delete icon to the email notification, allowing users to delete the email associated with the most recent notification directly from the notification rather than having to go to that email within the email app, deleting it there, and then returning to what they were doing (see Figure 9). This smoothing out of the users' activity flow is a good model for what actionable notifications should enable—quick, app-and-notification-appropriate actions that allow users to maintain "flow"[1] in their current context of activity on their device.

webOS 3.x's stacked notifications, while helpful, allowing limited triage of your notifications without leaving the larger context of the app you were in (as the Notification Dashboard is "merely" an overlay over your app), is still too limited in its usefulness. A more powerful way of managing your email notifications, for instance, would allow for users to take action on not only the notification, but on the email represented by the notification. But I'll leave further development of that idea to the Proposed Design section.

[1] I refer here to the notion of flow as developed by Mihaly Csikszentmihalyi in his *Flow: The Psychology of Optimal Experience*: "a state of concentration or complete absorption with the activity at hand and the situation" (from Wikipedia: http://en.wikipedia.org/wiki/Mihaly_Csikszentmihalyi)

CONFIDENTIAL



Figure 9 From webOS v2.2.4 (from the Pre 3). The webOS v2.2.4 version of the "Delete From Notification" patch. The added trash can icon allows users to delete the most recent email directly from the notification in the dashboard.

iOS v6.x Notifications: Playing Catch Up

It was not until 2011 with the release of iOS v5 and the introduction of iOS' Notification Center that iOS moved beyond an all-modal alert system, in which all alerts were modal. To illustrate, if a user were typing an email, and that user were to receive a text message, an alert about that text message would pop up, and that user would have to choose to dismiss that alert or go to see the text message in the messaging app. If they chose to dismiss that alert, that notification would be gone, leaving users to have to remember that that message had come in or to have to look for a badge on the messaging app's icon on their launcher screens.

With iOS v5, iOS gained a Notification Center and non-modal notifications. Notifications would pop up briefly at the top of the screen, and then disappear (Figure 10). Users who missed them or who intended to act on those notifications later would then have to remember to look in the Notification Center for them; the only other persistent visual cue that there were pending notifications were badges on launcher screen app icons. Nevertheless, the Notification Center provided a needed centralized location where users could look to find all their notifications. In practice, this approach scales more successfully than webOS' approach of one icon in the notification area per app-with-notifications. Given the number of apps available and potentially installed on each user's iOS device, webOS' approach might not be workable.

CONFIDENTIAL

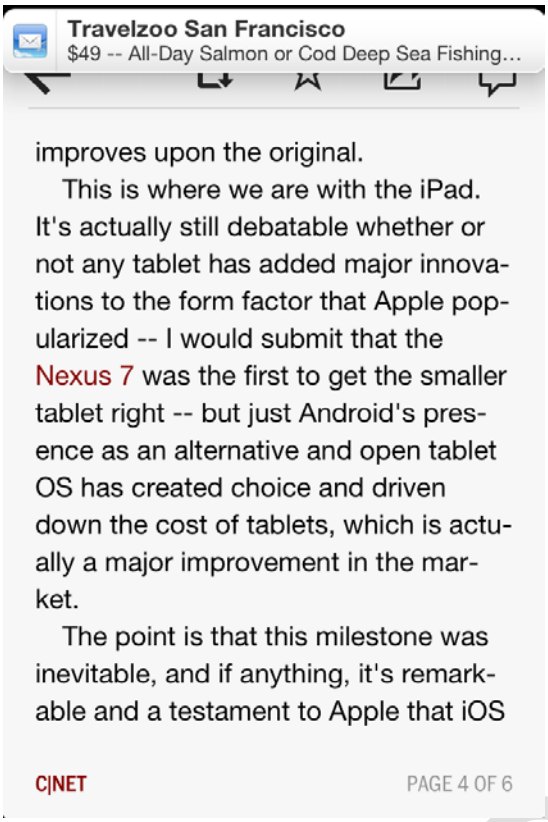


Figure 10 A banner notification from iOS v6.1.2. An incoming email from Travelzoo triggers a non-modal notification that appears at the top of the screen and then disappears after a few seconds.

With iOS v6, the Notification Center didn't see large UI changes, but it did gain the fantastic ability to sync notifications across iOS devices.[2]

The current iOS v6 notification system provides most of the benefits listed in the Overview:

- Unobtrusive notifications
- Persistence despite being ignored (rather than actively dismissed) by users, visible only by looking in the Notification Center
- Tapping the notification itself takes users to that notification's content within the relevant app.
- Users have very granular control over which apps can issue notifications, and which level of obtrusiveness they are allowed.
    - There are exceptions to this, however, and those exceptions can cause problems. One example of such is that phone call notifications stop any video recording activity, essentially behaving as a modal alert. I reported this behavior as an issue and used as a minor design exercise on my website on 2/7/2013, http://edwinli.com/ux/2013/02/07/incoming-calls-shouldnt-interrupt-video-recording-on-smartphones/.

[2] As reported in "iOS 6 Feature: Improved Notification System" on June 14, 2012: http://www.iphonehacks.com/2012/06/ios-6-feature-improved-notification-system.html

CONFIDENTIAL

Among the weaknesses of iOS' treatment of notifications:

- In the Notification Center, notifications are distinctly listed by app, but it is not possible to dismiss them individually, only as a group by app. This limits the utility of the Notification Center as a task list, as it is therefore not possible to mark off the tasks one by one as you might accomplish them.

Android v4.2 Jelly Bean: A Quick Look

I'm mostly familiar with Android's notification system as the system from which many believe that Apple drew inspiration for their own Notification Center. Android's Notification Drawer is also a panel that is summoned by a down swipe that lists all the undismissed notifications awaiting users' attention. While it most assuredly predates iOS' Notification Center, I don't know when it first appeared on Android, so do not know its temporal relationship with webOS' dashboard.

Android's notification system has been said to include:

- Unobtrusive notifications
- Persistence despite being ignored (rather than actively dismissed) by users, visible only by looking in the Notification Drawer
- Tapping the notification itself takes users to that notification's content within the relevant app.
- Android is known for its configurability. I do not have firsthand knowledge of how much control users have over which apps are allowed to issue notifications or what level of obtrusiveness they are allowed, but I expect that to be there. Further investigation will, of course, be necessary when undertaking the application of the design directions proposed in this document to improving Android's notification system.

New System Requirements

Goal of Phase One

In this first effort to improve notifications, the goal is to create notifications that better serve users, allowing them to triage their notifications and the tasks to which they lead while minimizing the disruption to their focus and flow in the context of the device activity in which they are currently engaged.

Notifications are the first step in potential multi-step process of doing an associated task. Until fairly recently, in most notification systems, the first step doing that task involved selecting a notification, then immediately leaving the app with which the user was recently engaged for whatever app is associated with the notification.

Some tasks, though, only take one or two steps, are common, and thus, not "worth" the flow disruption of switching app contexts.

CONFIDENTIAL

Thus, the goal is to offer as many actions as possible early enough in the task completion process to allow users to complete those common tasks while minimizing the disruption of their activity flow, or context. This seems to be the crux of the idea of "actionable notifications."

Inspiration: "Delete From Notifications" Patch (webOS)

As suggested above, one of my first experiences with the notion and benefits of actionable notifications comes from my experience with Palm's Pre smartphone running webOS v1.x. To reprise that discussion, the webOS Homebrew developer known as Lclarkjr on webOS Nation's forums[3], created a patch that added the ability to delete an email directly to the email dashboard notification.

Prior to installing the patch, if you were engaged in another app and you saw a notification for an email that you wanted to delete, you would have to go tap the notification in your Dashboard, which would switch you into the email app, opened to that particular email, delete the email there, then switch back to the app you were using before if you wanted to continue with what you were doing.

With the patch installed, the task of deleting that email was much shorter and less disruptive: open the Dashboard, skim the info provided from the email (From address and subject line), tap delete, tap on your main app (still very visible), and continue your activity.

This was a much smoother process, allowing users to "get things done" with respect to their task list as represented by their notifications list while still maintaining the flow of their other activity.

The patch left me hoping for other expansions on this idea, with other common actions, and use of the idea beyond just the email app.

It was some years of webOS use before I saw another suggestion of that, and it was in using the toodleTasks HD app by ProdiSoft (http://www.prodisoft.ch/), a task management application for use with the website service ToodleDo. The webOS app's dashboard notification included not only the ability to go directly to the task item in the app, but also to postpone the task or to mark it complete (Figure 11). Although the exact behavior of "postpone" is a little unclear (i.e., whether it applies to the notification or the task it represents), the addition of those actions to the notification is an advance. Disappointingly, the "complete" link switches users into the toodleTasks HD app, where the app marks the task as complete and then leaves users in the app. This app switching probably has more to do with limitations of webOS with respect to what can be passed from a notification to the app in the background than with ProdiSoft's design goal here.

So toodleTasks HD's dashboard notification is suggestive, but ultimately doesn't quite meet our goals here.

[3] http://forums.webosnation.com/webos-patches/215142-patch-email-delete-notification-bar.html

CONFIDENTIAL

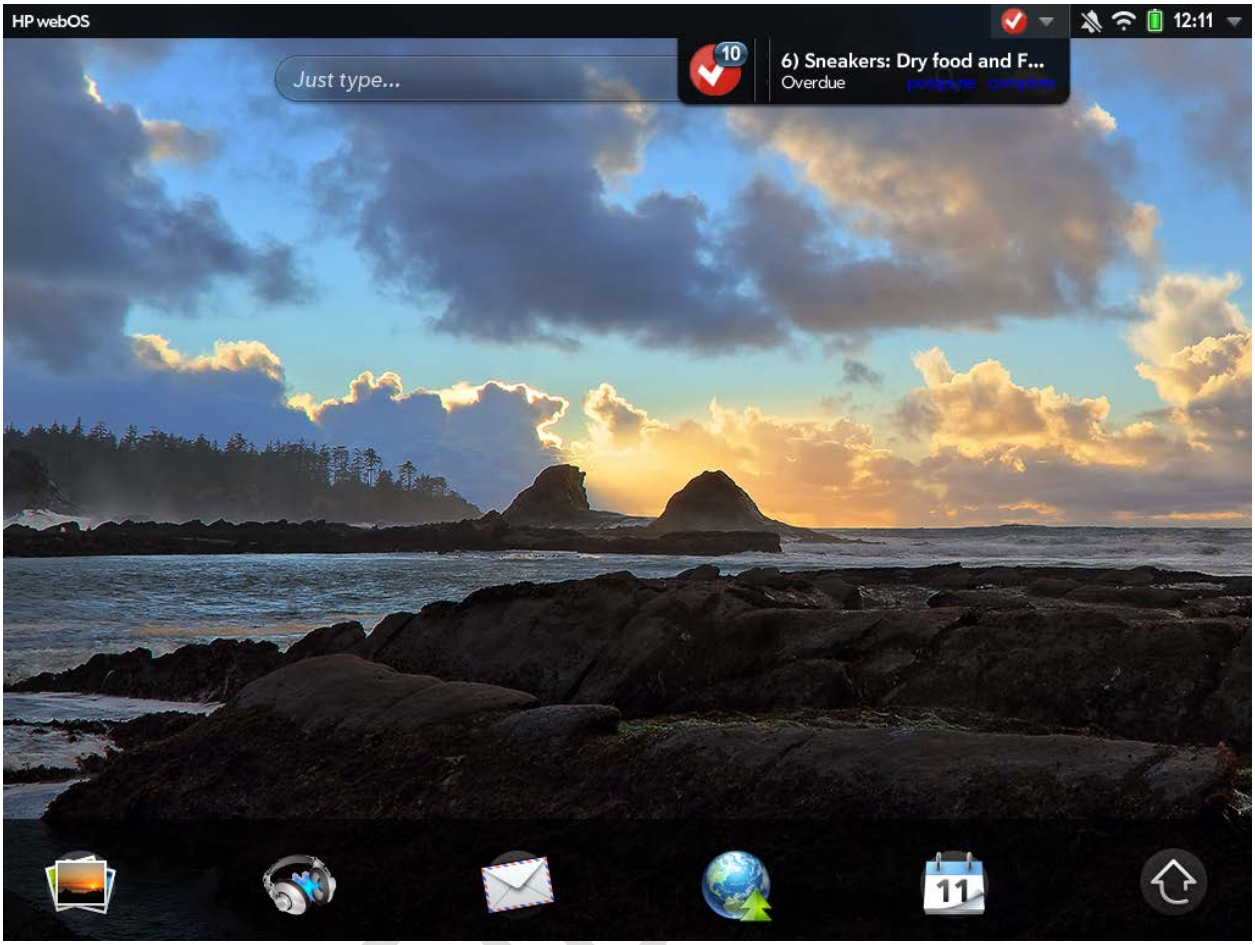


Figure 11 Dashboard notification from app toodleTasks HD (by ProdiSoft, http://www.prodisoft.ch/), from webOS 3.0.5 (from an HP Touchpad). Tapping on the icon leads to the app, opened to its main screen. Tapping on the task text ("Sneakers…") leads to the task item in the app. The two blue links, "postpone" and "complete" are actions that one can invoke from the notification. Sadly, though, the "complete" link actually switches users' context into the app, where it automatically marks the relevant app complete.

Beyond that, still to be done with respect to webOS as an example would be to encourage the use of this idea by creating a standard way to offer these actions in notifications OS-wide.

Inspiration and Framework: Getting Things Done

Getting Things done is a popular time management methodology laid out in the book titled "Getting Things Done: The Art of Stress-Free Productivity" (by David Allen). Its main tenets are to record everything that you're trying to keep track (especially tasks) of into a single system, building lists of tasks and a system for managing them, a classification system for tasks, and rules about how to deal with tasks.

Critically, according to the book, the recording of everything frees your mind from having to keep track of them, allowing you to focus on the activity before you and maintain a state of flow in that activity.

Of the rules that Allen uses, two are of particular interest here. A two minute rule for tasks allows the doing of tasks "now" if they should take less than two minutes to do, with the primary notion being that

two minutes is a short enough time to allow you to maintain focus and flow in the context of your larger activity. The other rule lists four options in dealing with a task: do it, delegate it, defer it, or drop it.

Considered in this framework, notifications are task items to which users must attend, and notification lists—whether webOS' Notification Area, Android's Notification Bar, or iOS' Notification Center—are lists of tasks.

To provide only the two options to either dismiss or attend to a notification is to offer only the task options of dropping it or doing it. Further, since attending to a notification means leaving the context of your activity (i.e., whatever app you're in) to go into that notification's app, it means disrupting your focus and flow.

This suggests two directions for evolving the design of a notification system:

1. To offer notification choices which are equivalent to deferring a task and delegating a task.
2. To offer ways for users to "get the notification done" while minimizing or eliminating the degree of contextual change, thus minimizing the disruption to users' focus and flow.
    a. Consider here, too, that some tasks indicated by a notification may be simple, requiring less than the full UI of its associated app, or it may be more complicated, requiring the app's full UI.

These directions strongly suggest a movement towards "actionable notifications" and have implications for wider notification system changes. The first phase of notification system improvement documented here will focus on the first of these, leaving wider notification system changes to a later phase.

Importantly, GTD may not be a methodology that fits all users, so any evolution of a notification system should minimize the degree to which users might have to adjust if they want to maintain their old behaviors with in the newer system.

Parallel Work

I'm not very familiar with Android, but their What's New page, featuring version 4.2 Jelly Bean, refers to "Actionable Notifications":

> Android has always put you in control when it comes to staying notified and connected.
> Just swipe down from the top of the screen to see all your notifications in one place.
> Late for a meeting or missed a call? Take action in an instant directly from the notifications shade.
>
> (from http://www.android.com/whatsnew/)

CONFIDENTIAL



Figure 12 Android v2.4 Jelly Bean's picture of an "actionable notification," from their What's New page, http://www.android.com/whatsnew/.

CONFIDENTIAL

The current version of the Gmail app for Android also has actionable notifications. Reported on March 18, 2013 by Eric Ravenscraft at AndroidPolice.com (http://www.androidpolice.com/2013/03/18/gmail-update-allows-you-to-reply-archive-or-delete-emails-right-in-the-notification-shade/), a Gmail notification in the Notification Drawer gives users the choice to go directly to the email, to archive the email (gmail's preferred action to deleting emails), or reply to the email.

This is very much in line with the design direction that my design proposes, bringing the most common email-specific actions from the app essentially as far up in the "task process" to its beginning in the notification as possible without bringing the full UI of that notification's app to the forefront and forcibly supplanting the context in which the user was already in.

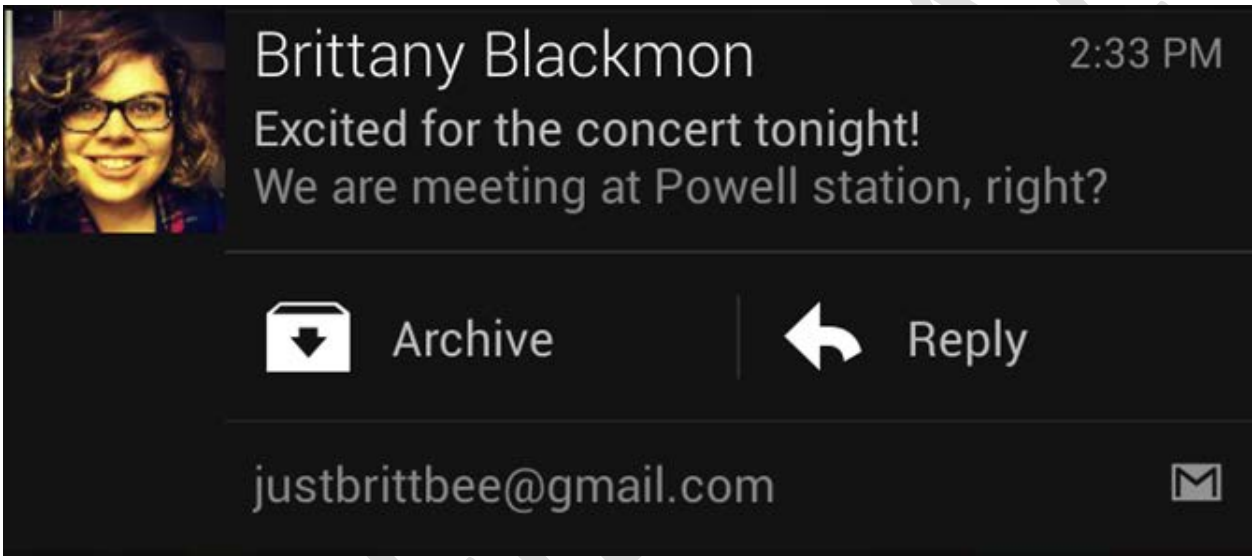


Figure 13 Notification from Gmail app v4.3 for Android, taken from an Android device running v4.2 Jelly Bean. On 3/18/2013, Eric Ravenscraft reported that the new Gmail app offers a new action available from its notification: archiving that particular email. Picture reproduced from http://www.androidpolice.com/2013/03/18/gmail-update-allows-you-to-reply-archive-or-delete-emails-right-in-the-notification-shade/.

Process

Even as a personal project, the creation of this mobile app will follow a user experience-focused process, incorporating best practices and iteration to create and develop a design to improve notifications that will be increasingly useful, usable, delightful, and flow-facilitative.

Here are steps along the journey:

1. **Domain research.** Are there aspects of the domain that may be identified as domain structure or constraints that may influence or be taken advantage of in the ultimate design?
2. **User behavior research.** How can users current behavior in the current domain-state be described? Does that behavior suggest positives or negatives with respect to accomplishing their goals within the domain? Are there regularities that may be taken advantage of when positive or done away with when negative?

3. **Market research.** Are there solutions already out there? What's out there that is related? How well do they support users in their activities and goals? How are they strong or weak with respect to your design and experience goals?
4. **Digesting the data into design-useful artifacts.** This process may include diagramming, card sorting, re-representing the notes and data in various ways. Products of this process may include diagrams, personas, scenarios, sketches, lists of problems, lists of requirements, etc. This is a quote by Herbert Simon of particular relevance to me at this stage: "solving a problem simply means representing it so as to make the solution transparent." As such, this stage may very well blend and flow back and forth with the next stage, Design.
5. **Design.** Of course, this is an iterative and circular process with sub-processes. It can involve casting about for inspiration and models, lists and re-lists of requirements, content models, navigation models, task models, sketches, Balsamiq mockups, HTML, even Photoshop--whatever's quick, easy, easily let-go, and supportive of reflection.
    a. Sub-processes: Idea generation, Idea exploration, Make choices and "go deep", then repeat recursively as design problems are identified.
6. **Develop.** This will interact with the Design processes as constraints and interactions between elements are identified.
7. **Usability/User Experience testing.** What's it like to use the product at its various stages of being designed and developed? It should be tested, evaluated all along the way, with lessons learned from testing adding to the mix of Idea generation, Idea exploration, "going deep", etc.
8. **Wrap up version and release…**
9. **Repeat the processes looking towards future versions.**

Process Steps Represented In This Document

This document represents a snapshot of the state of the described Location-Based Classified Ad Mobile App in terms of the process outlined above and how it got there.

1. **Domain research.** Done as a user of webOS and iOS. While self-observation and introspection can often be suspect as a source of research, as a personal project and to limit the scope of work go get to the design phase, I judge this acceptable. **Artifacts:** notes and documentation (current document).
2. **User research.** As with domain research, the data collected here comes from self-observation and introspection. **Artifacts:** notes, lists of tasks, lists of goals, lists of problems, requirements, etc.
3. **Market research.** Investigating and elucidating how notification systems work on webOS, iOS, and Android (incomplete). **Artifacts:** this document.
4. **Digesting the Data. Artifacts:** lists of requirements, problems, potential solutions, functionality in service to problems, etc.
5. **Design.** Part of the design comes from the inspiration/choice of a mobile platform/app as the basis of the solution. Further design constraints and structure come from the choice of HP

CONFIDENTIAL

webOS as the platform of choice. **Artifacts:** content models, navigation models, sketches, Balsamiq mockups.

This is as far as my current process has gotten, and I am still in the design and prototyping stages, with forays "back" into market research and digesting data.

Artifacts Included in This Document

Included in this document are sample artifacts from the unfinished process of designing improved notifications in webOS:

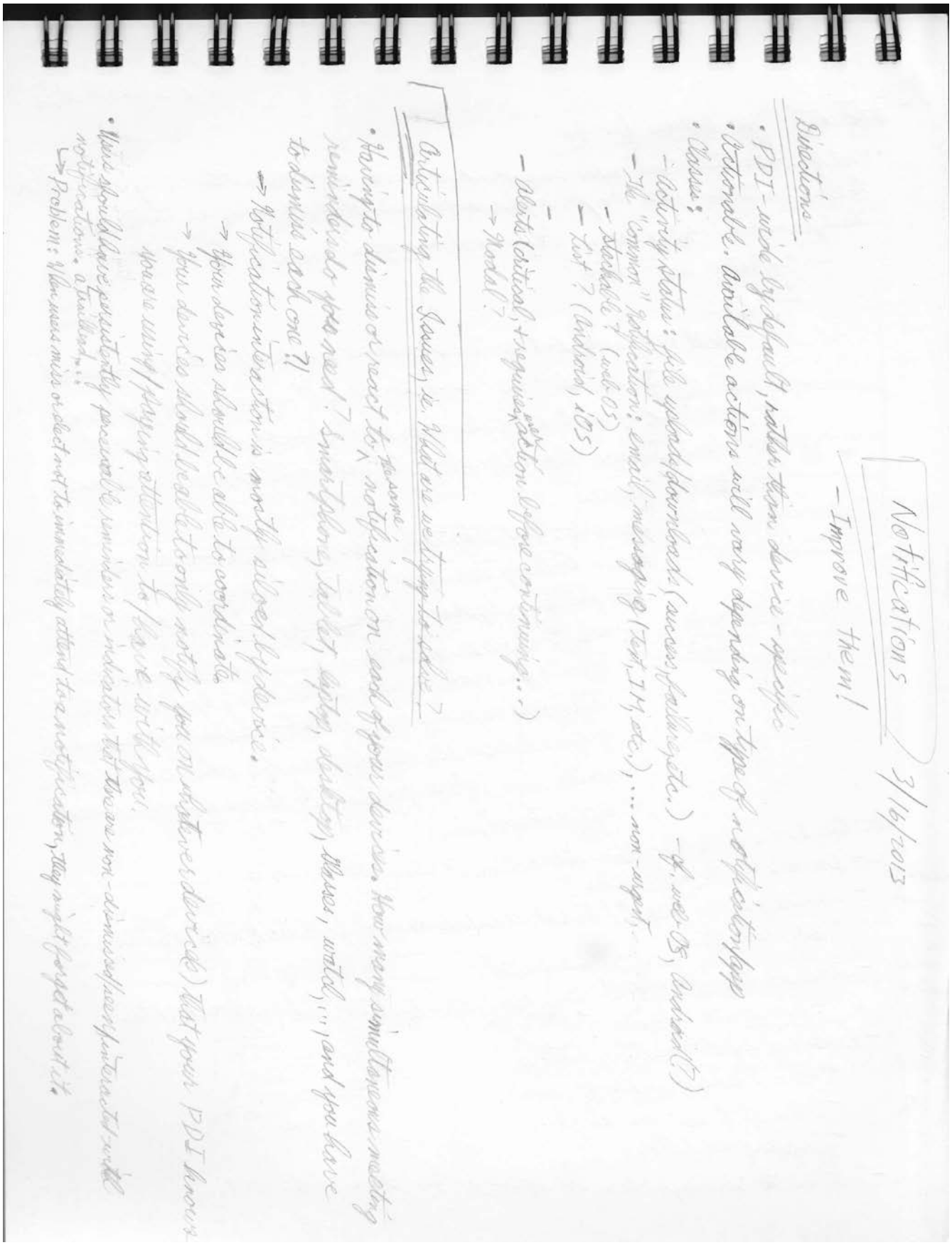
- Notes on the problem space, design considerations, and design directions.
- A listing and sketching of current-state webOS notification states.
- Sketches of design directions. Where to put the actions, how to display them, challenges in the use of icons, and how to obviate those challenges.
- Thinking ahead to Phase Two: how might a notification system work in the context of an ecosystem/Personal Digital Infrastructure? A scenario sketched out allowing a user to assign a notification or task to a particular device's notification or task list, exploring the notion of each device having its own task list.
- Mockups created in Balsamiq Mockups, exported as PDF pages.

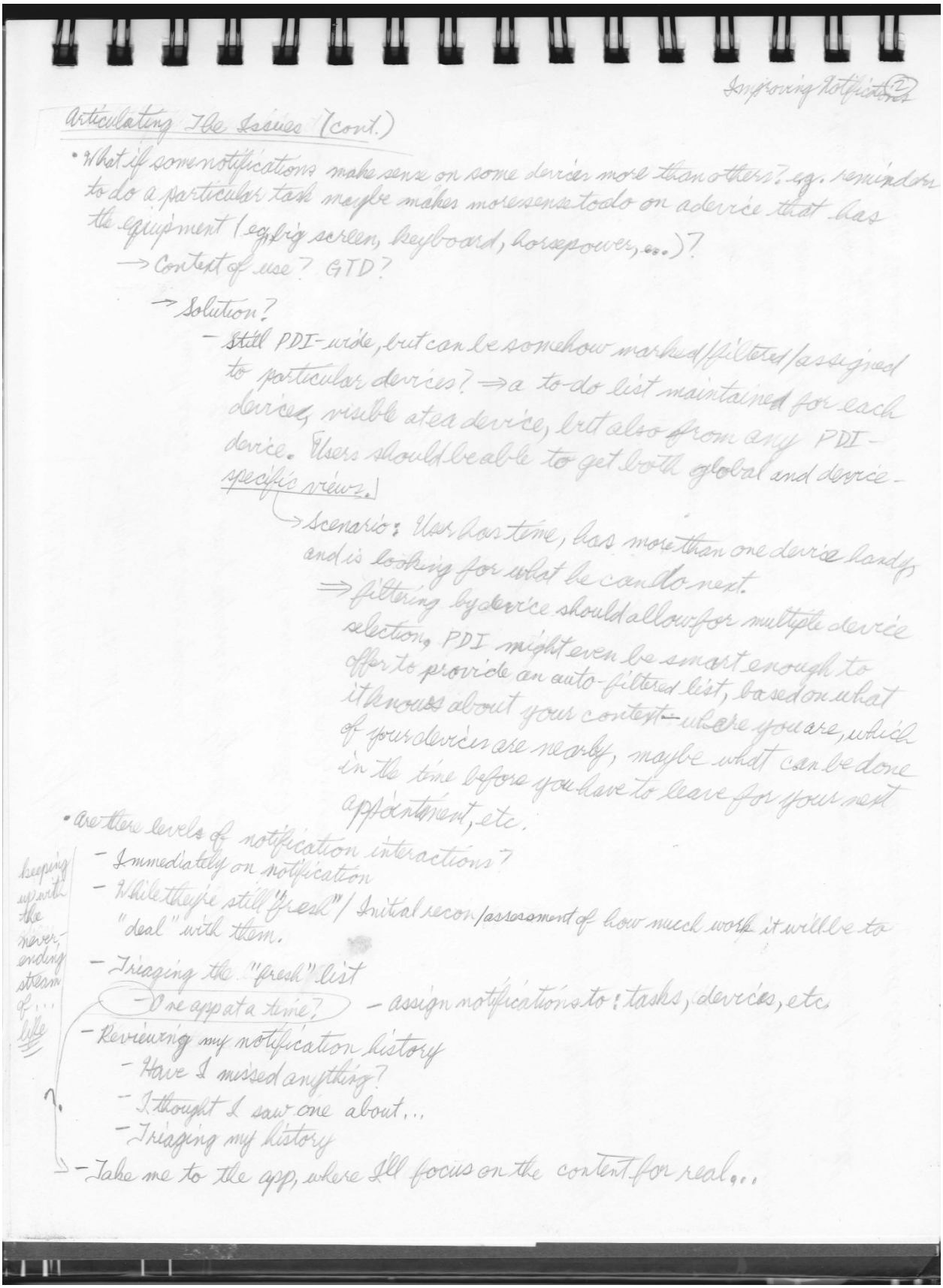
CONFIDENTIAL

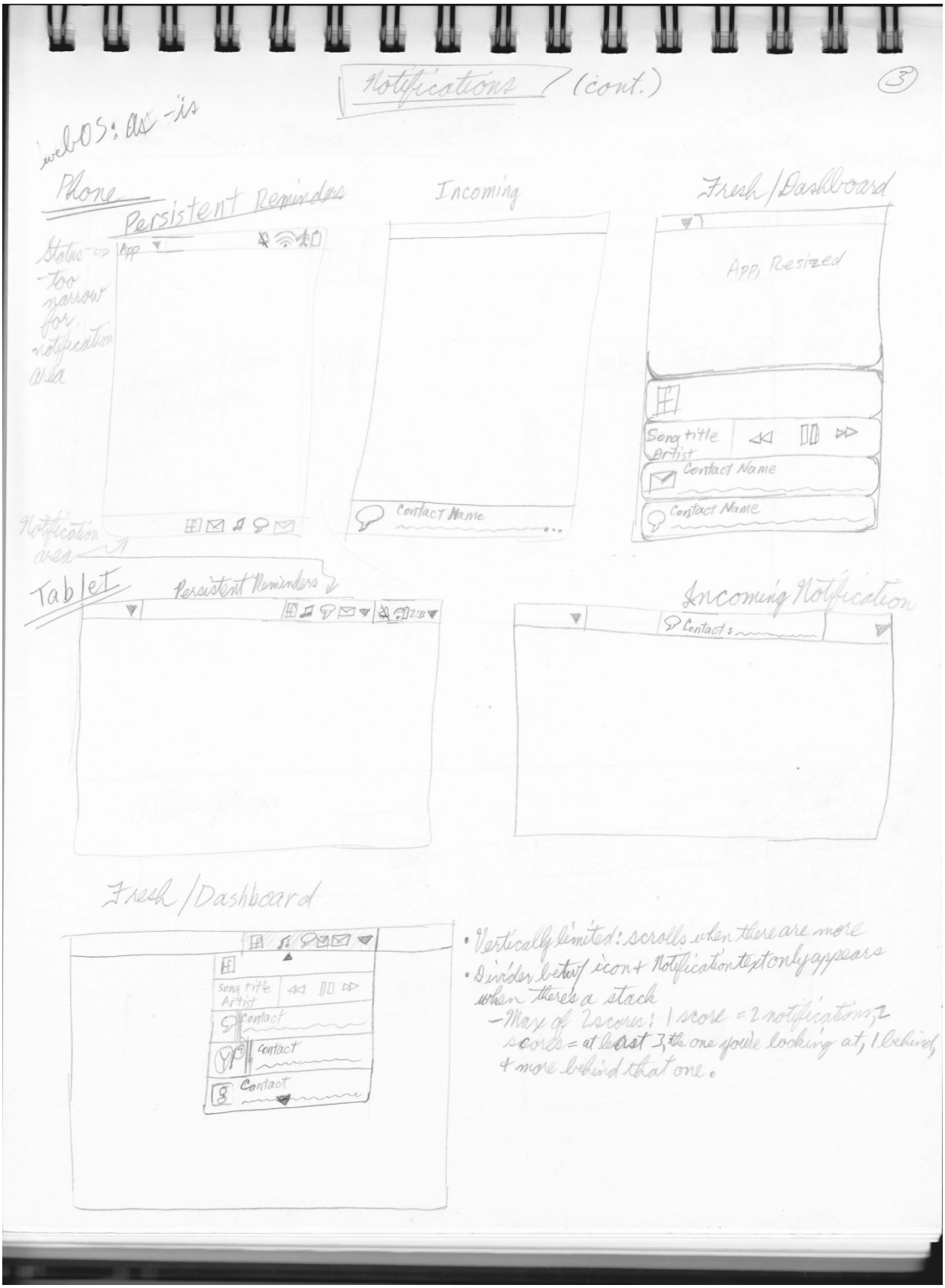
Data, Digesting the Data, and The Beginnings of Design

In this section: Notes on design directions and considerations for both phase one and phase two, and sketches

CONFIDENTIAL


Notifications
3/16/2013
- Improve them!
Directions
• PDT-wide by default, rather than device-specific
• Actionable. Available actions will vary depending on type of notification/app
• Cases:
- Activity status: file upload/downloads (success/failure/etc.) - ex. web UI, Android(?)
- The "common" notification: email, messaging (text, IM, etc.), ... non-urgent
- Actionable? (web OS)
- Ext? (Android, iOS)
-
- Alerts (critical, requiring action before continuing...)
→ Modal?
Articulating the Issues to Address
• How to dismiss or react to/resume a notification or task of your device. How may simultaneous meeting reminders you need? Smartphone, tablet, laptop, desktop, phones, watch, ..., and you have to dismiss each one?!
⇒ Notification interaction is mostly siloed by device.
→ Your laptop should be able to coordinate
→ Your device should be able to notify you on whatever device (not your PDT) knows you're using/paying attention to/are likely to see it.
• We should have proactively provide reminders or indicators for those non-dismissed/unattended notifications. Evolution...
↳ Problem: When users miss or elect not to immediately attend to a notification, they might forget about it.


Improving Notifications (2)
Articulating the Issues (cont.)
• What if some notifications make sense on some devices more than others? eg. reminders to do a particular task maybe makes more sense to do on a device that has the equipment (eg big screen, keyboard, horsepower, ...)?
→ Context of use? GTD?
→ Solution?
– Still PDI-wide, but can be somehow marked/filtered/assigned to particular devices? ⇒ a to-do list maintained for each device, visible at ea device, but also from any PDI-device. Users should be able to get both global and device-specific views.
→ Scenario: User has time, has more than one device handy, and is looking for what he can do next.
⇒ filtering by device should allow for multiple device selection. PDI might even be smart enough to offer to provide an auto-filtered list, based on what it knows about your context — where you are, which of your devices are nearby, maybe what can be done in the time before you have to leave for your next appointment, etc.
• Are there levels of notification interactions?
– Immediately on notification
– While they're still "fresh" / Initial recon/assessment of how much work it will be to "deal" with them.
– Triaging the "fresh" list
– One app at a time?
– assign notifications to: tasks, devices, etc
– Reviewing my notification history
– Have I missed anything?
– I thought I saw one about...
– Triaging my history
– Take me to the app, where I'll focus on the content for real...
keeping up with the never-ending stream of... like


Notifications (cont.)
3
webOS: as-is
Phone
Persistent Reminders
Incoming
Fresh/Dashboard
App
Status too narrow for notification area
Notification area
Song title
Artist
Contact Name
App, Resized
Tablet
Persistent Reminders
Incoming Notification
Contact:
Fresh/Dashboard
Song title
Artist
Contact
Contact
Contact
• Vertically limited: scrolls when there are more
• Divider betwn icon + Notification text only appears when there's a stack
– Max of 2 scores: 1 score = 2 notifications, 2 scores = at least 3, the one you're looking at, 1 behind, + more behind that one.

Design

In this section: Documentation, Notes, sketches, and Balsamiq Mockups.

Design Directions

To collect and reiterate points discussed earlier, the design directions for this first phase of improved notifications can be described as creating an actionable notification design.

Possible directions include:

1. To offer notification choices which are equivalent to deferring a task and delegating a task (in GTD terms)
2. To offer ways for users to "get the notification done" while minimizing or eliminating the degree of contextual change, thus minimizing the disruption to users' focus and flow.
    a. Consider here, too, that some tasks indicated by a notification may be simple, requiring less than the full UI of its associated app, or it may be more complicated, requiring the app's full UI.

The goal is to offer as many actions as possible early enough in the task completion process to allow users to complete those common tasks while minimizing the disruption of their activity flow, or context. This seems to be the crux of the idea of "actionable notifications."

Choice of Operating System and Device

For this design exercise, I have chosen to base my design on that of HP webOS' well-regarded notification system (primarily the Touchpad, running webOS version 3.0.5) due to my personal familiarity with it. The HP Touchpad is a capacitive touchscreen tablet with various combinations of wireless technologies and sensors, including wi-fi and Bluetooth.

The choice of operating system is the choice of how to express a design idea in a particular "idiom". A design idea explored in webOS can then be taken and adapted or applied to any other mobile OS, such as iOS, Android OS, Windows Phone, Blackberry OS, or Firefox Mobile OS.

CONFIDENTIAL

Improved Notifications, v0

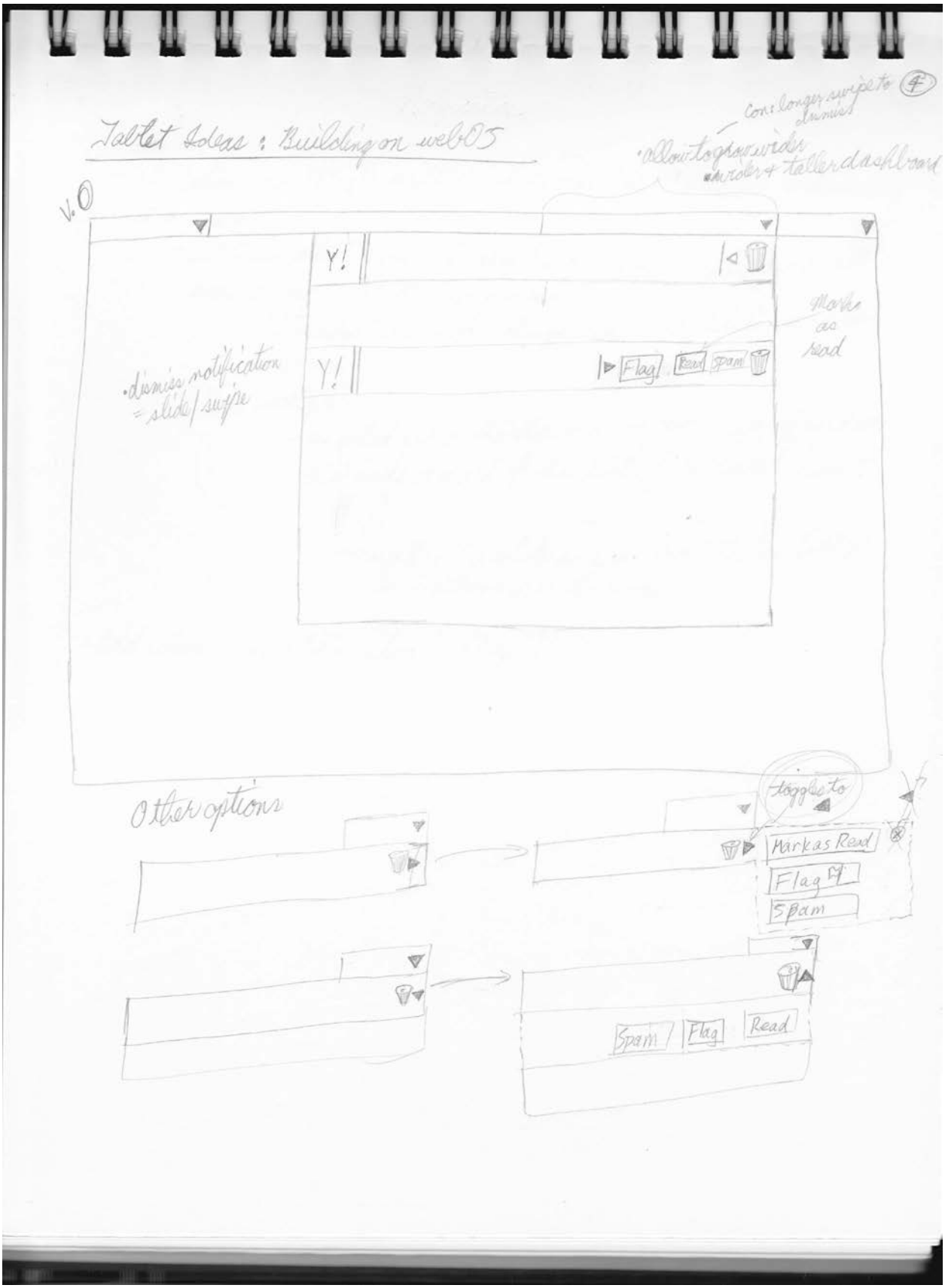
In this section: Documentation, Notes, sketches, and Balsamiq Mockups.

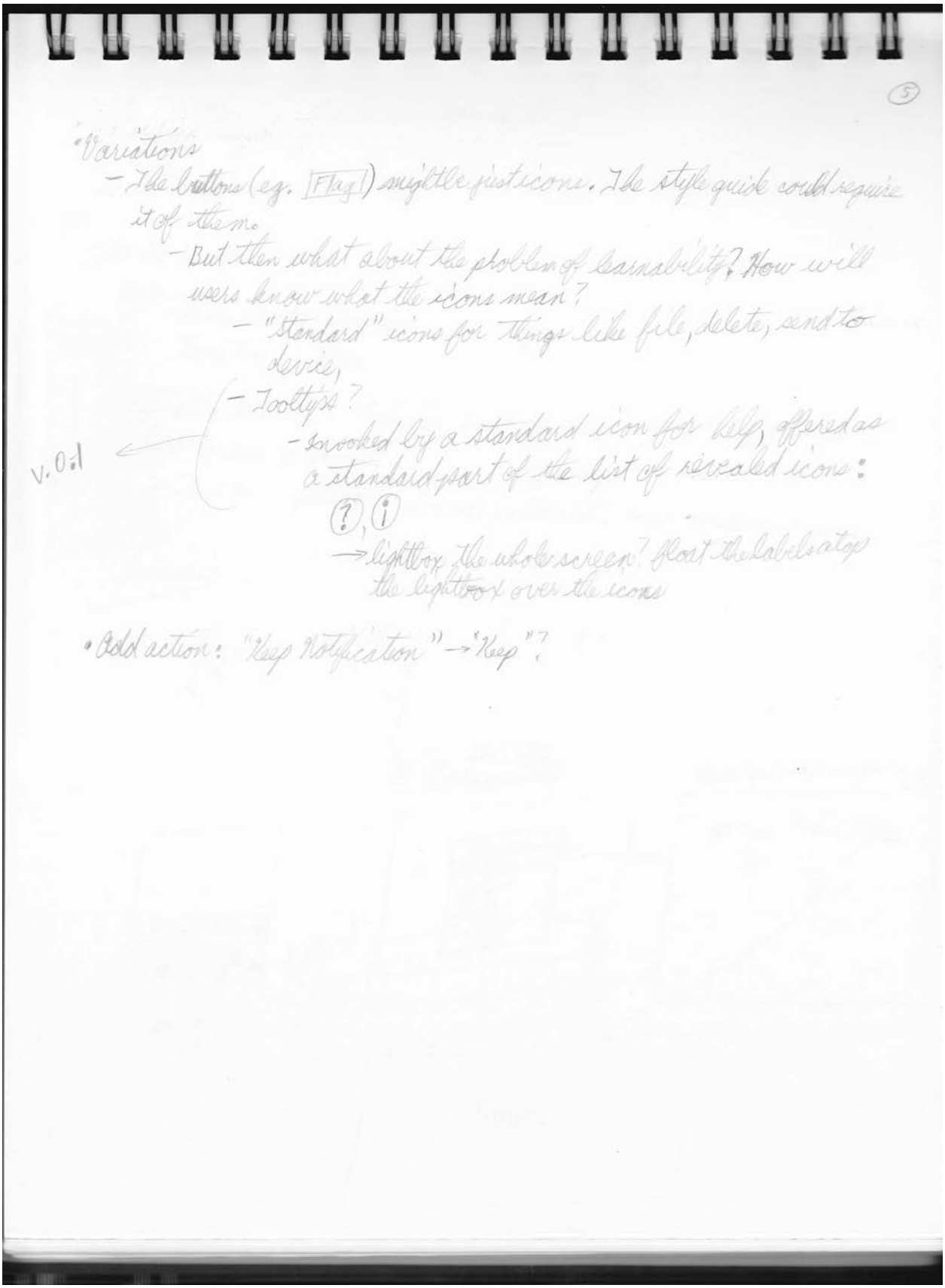
Exploring More Than One Design

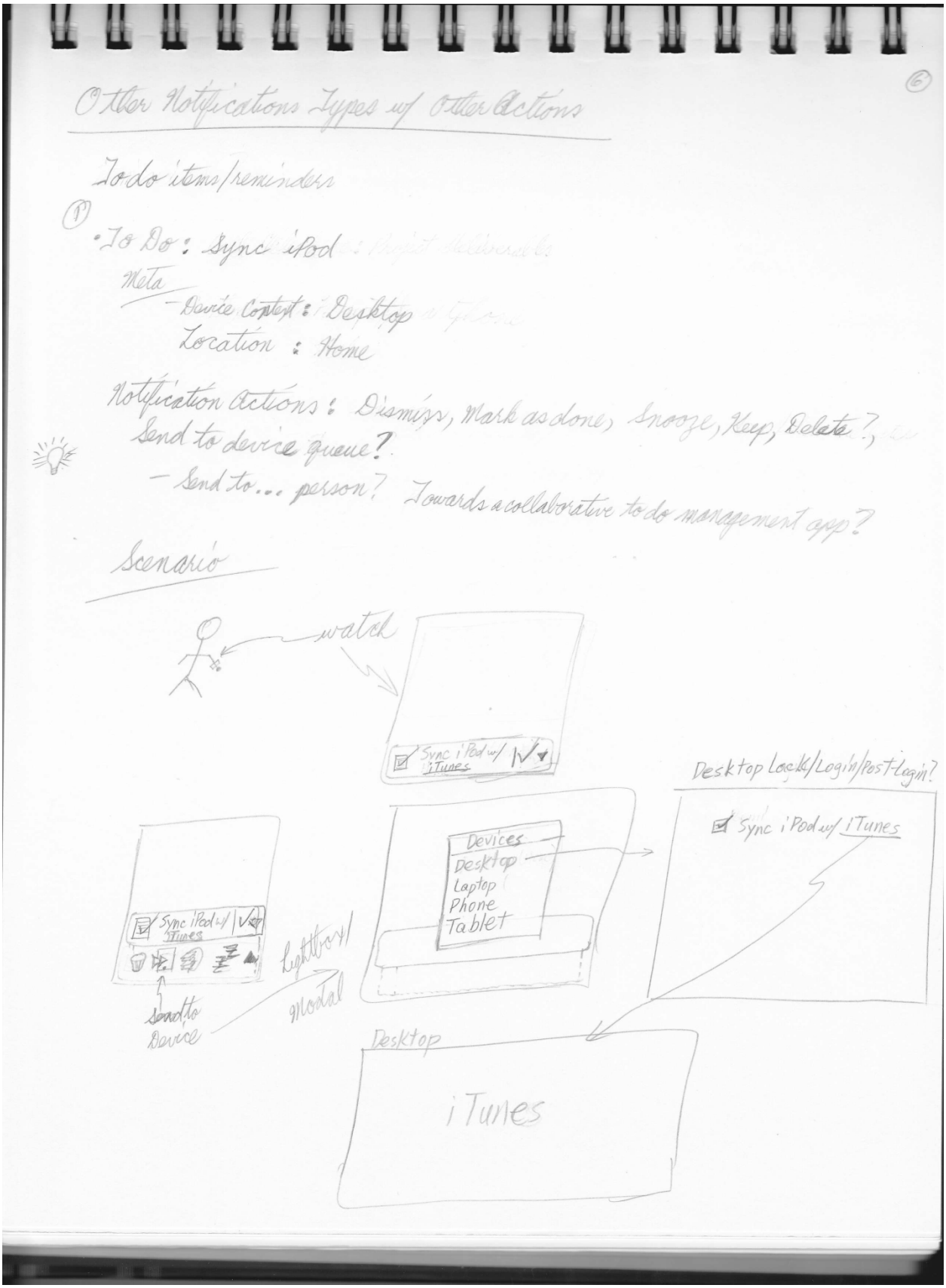
The sketches and Balsamiq mockups explore a few different approaches to the same solution of providing an action drawer (called a command drawer in many of the mockups) for the notifications, most of the variation having to do with placement of the drawer…as an overlay on the notification, to the right of the notification, and below the notification.

Many of my thoughts and reactions along the way are documented as annotations on the sketches or in comment boxes in the Balsamiq mockups.

CONFIDENTIAL


Tablet ideas : Building on web05
V.0
Y!
Y!
Flag
Read
Spam
Mark as read
•dismiss notification = slide/swipe
Other options
toggle to
Mark as Read
Flag
Spam
Spam
Flag
Read


5
•Variations
– The buttons (eg. Flag) might be just icons. The style guide could require it of the mo
– But then what about the problem of learnability? How will users know what the icons mean?
– "Standard" icons for things like file, delete, send to device,
– Tooltips?
– invoked by a standard icon for help, offered as a standard part of the list of revealed icons:
v. 0.1
?, i
→ lightbox the whole screen? float the labels atop the lightbox over the icons
• Add action: "Keep Notification" → "Keep"?


Other Notifications Types w/ Other Actions
To do items/reminders
• To Do: Sync iPod
Meta
– Device Context: Desktop
Location: Home
Notification actions: Dismiss, Mark as done, Snooze, Keep, Delete?
Send to device queue?
– Send to... person? Towards a collaborative to do management app?
Scenario
watch
Sync iPod w/ iTunes
Desktop Lock/Login/Post-login?
Sync iPod w/ iTunes
Devices
Desktop
Laptop
Phone
Tablet
Sync iPod w/ iTunes
Send to Device
Lightbox/ modal
Desktop
iTunes

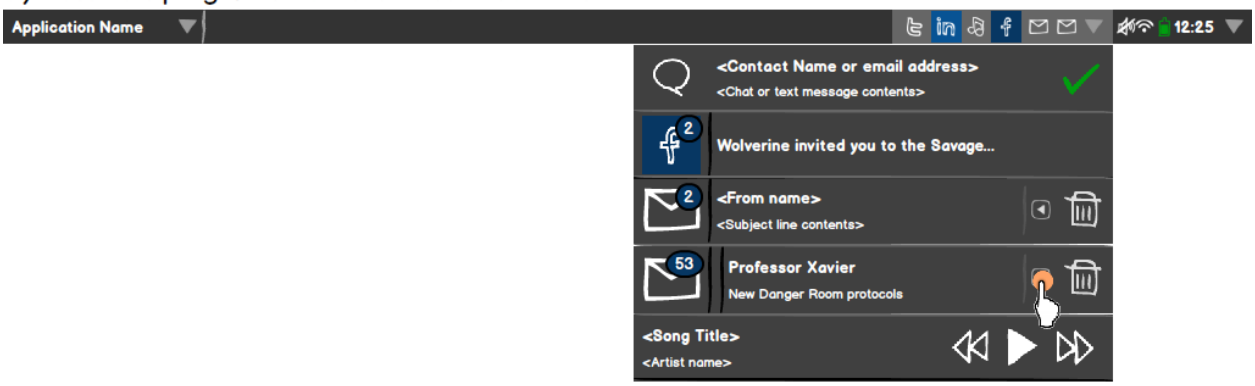
Improving Notifications v0, webOS-Style: Adding an action drawer

Design v0a: Action drawer as an overlay

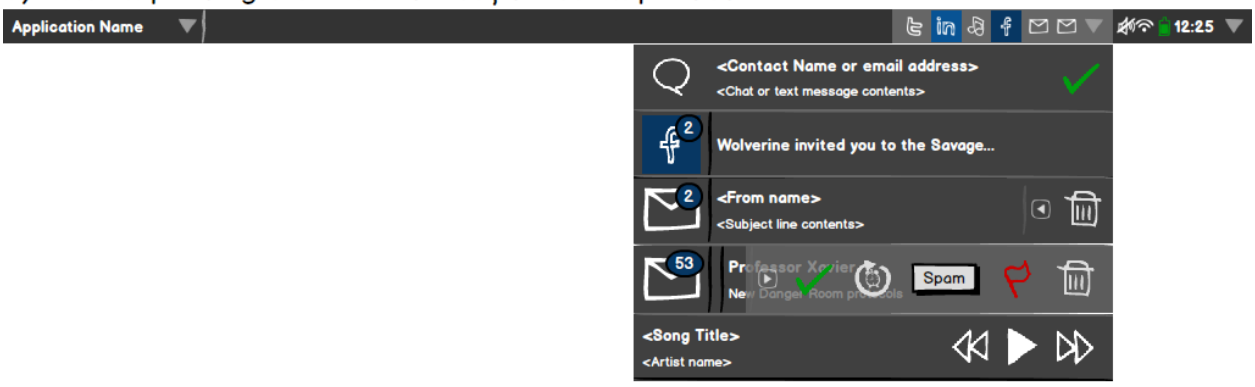
1) Status bar from within an app.



2) After swiping down on the notification area



3) After expanding the action drawer, drawer expanded



Could be difficult for users to read. Too many layers visible through the various layers of transparency? Maybe make the drawer 100% opaque, i.e., no transparency at all.

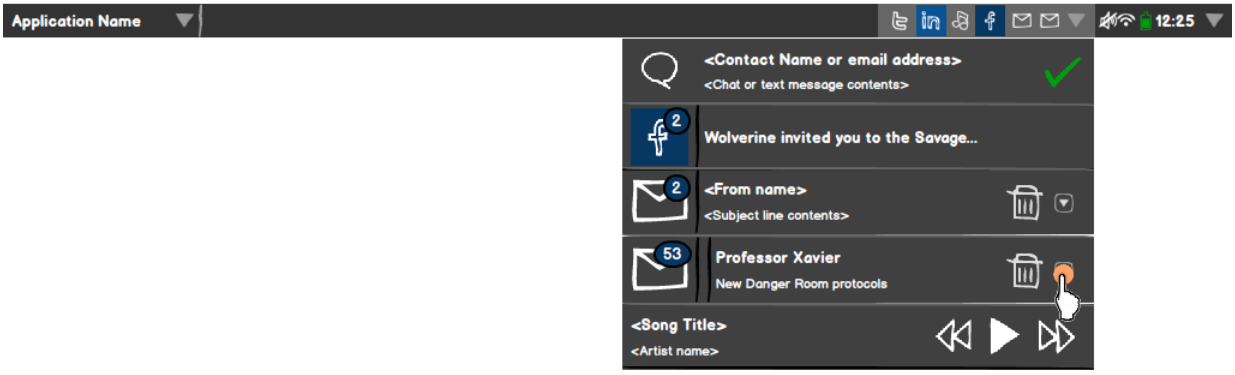
Improving Notifications v0, webOS-Style: Adding an action drawer

Design v0b: Action drawer opens below notification

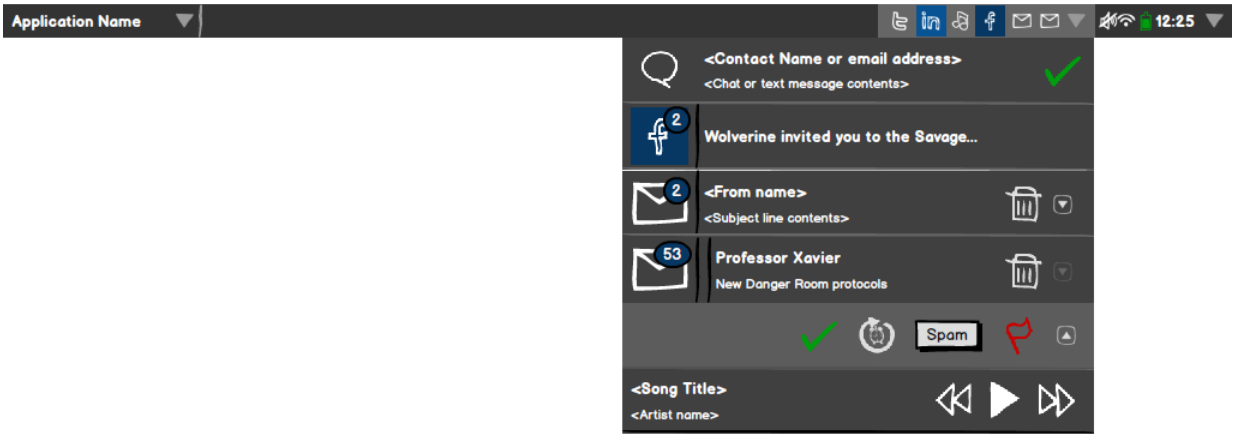
1) Status bar from within an app.



2) After swiping down on the notification area



3) After expanding the action drawer, drawer expanded



Expanding below the notification brings up a couple of issues.

1) Animatng the notifications below the notification of interest may look ugly or be disconcerting when it's done quickly enough not to be a pain over repeated usage. How quickly is quickly enough to not be a pain?

2) When the notifications are at the maximum height allotted for the notifications dropdown, introducing the drawer will push one notification below the "frame". This isn't a huge problem, knowing that the frame is already set up for scrolling and to indicate the need for scrolling (the triangles that appear at the top and bottom of the frame).

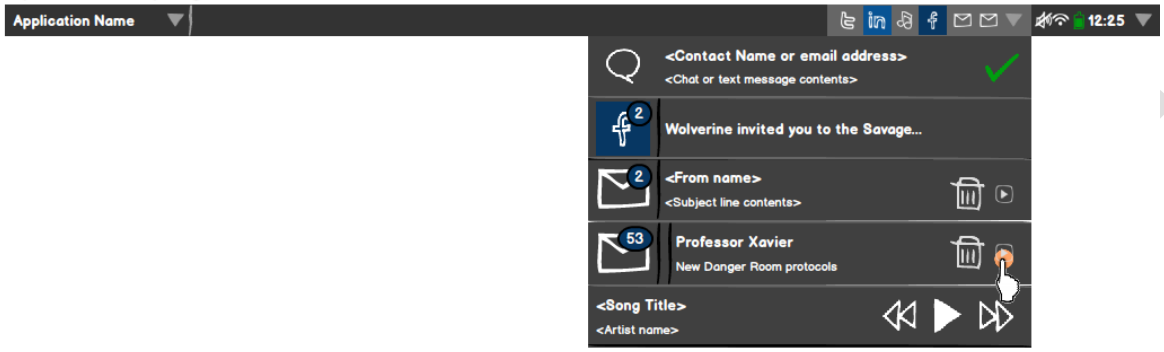
Improving Notifications v0, webOS-Style: Adding an action drawer

Design v0c: Action drawer opens to the right of the dashboard frame

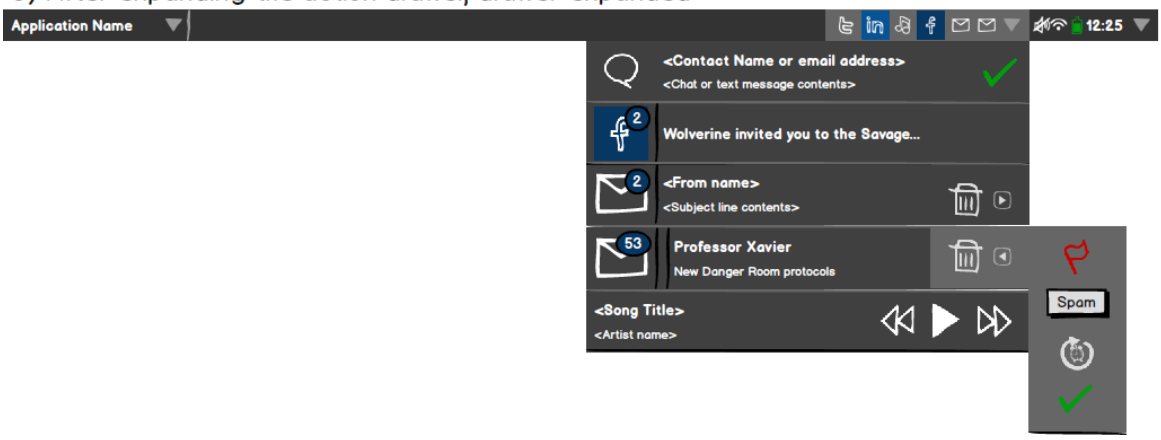
1) Status bar from within an app.



2) After swiping down on the notification area



3) After expanding the action drawer, drawer expanded



Issues With Expanding Right

1) The top or bottom of the drawer. It will sometimes bump up against the top or bottom of the notifications dropdown frame, especially when there are only one or a few notifications. Which means defining different behavior for different cases, which might be visually disorienting for users:

a) When near the top, the top of the drawer will have to align against the bottom of the status bar. When this is the case and there are only a small number of the notifications, the bottom of the drawer will have to be allowed to extend below the notification frame.

b) When the relevant notification is near the bottom of the list, whether at the bottom of the frame or not, it is tempting to align the bottom of the drawer to the bottom of the notifications dropdown. But that can't work...not as a single rule...

c) What if the drawer's top always aligns with the top of the relevant notification, such that it always drapes down? This leads to the most thoroughly predictable behavior.

d) Trying to center the drawer on its notification is nice, but it's the most problematic, running into problems at the top and the bottom of the notification frame. The attempt here to order the commands according to usage frequency from the middle is also problematic.

Answer seems to be C.

Improved Notifications v0.1

Considerations From v0

- Icons or buttons with text labels or both?
- If the design standard were to limit app designers to using icons, how do we help users understand icons whose meaning not clear?
    - InfoTips? Tooltips that give the text for the actions. But how to invoke them? ➔ A standardized “help”/InfoTips icon as an always-present member of the complement of icons.

CONFIDENTIAL

Improving Notifications v0.1, webOS Style

About v0 vs. v0.1

Whichever of the previous alternatives for the command drawer turns out to be best, they all rely mostly on icons, even if I choose to allow buttons. This brings up two issues:

1) If there are no buttons, the meaning of icons may not be clear. How do we make that clear?
2) If there **are** buttons, horizontal space will be problematic for label strings that require wider buttons. This is a problem whether the command drawer is horizontal or vertical/next to the notifications frame, but it is worse for the vertical version. This suggests the choice to restrict the command drawer to containing just icons.

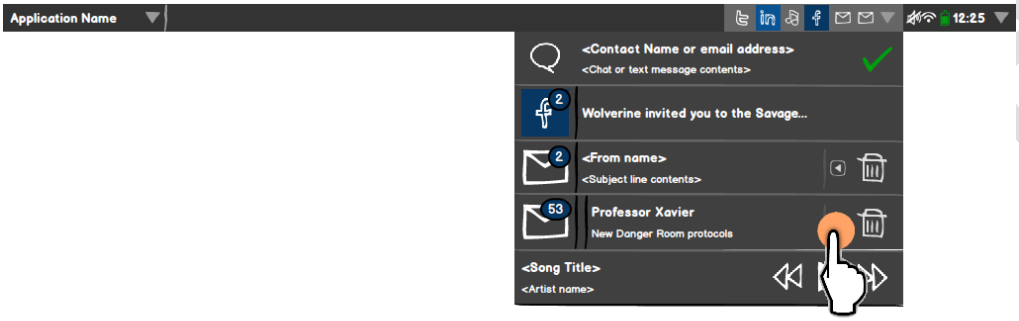
Between these two issues, one solution that presents itself is restricting the command drawer to icons, but adding in a standard icon to invoke "help" in the form of simultaneous "tooltips", such that small, left-aligned text labels appear over each icon. Those labels could allow for multiple lines of text, or just really long labels.

Design v0.1a: Action drawer opens as an overlay over the notification

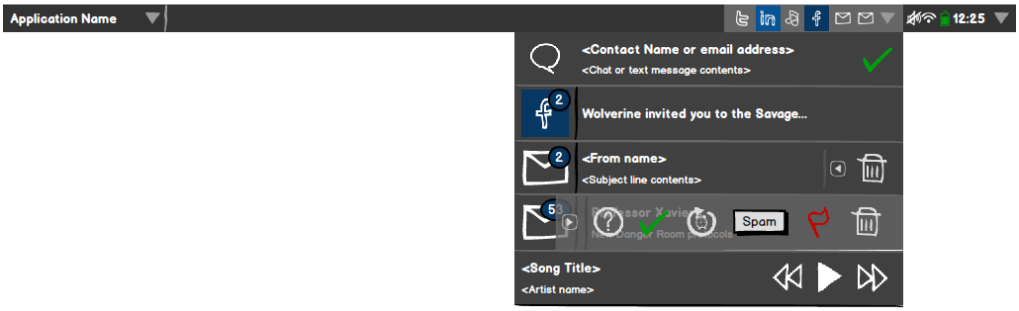
1) Status bar from within an app.



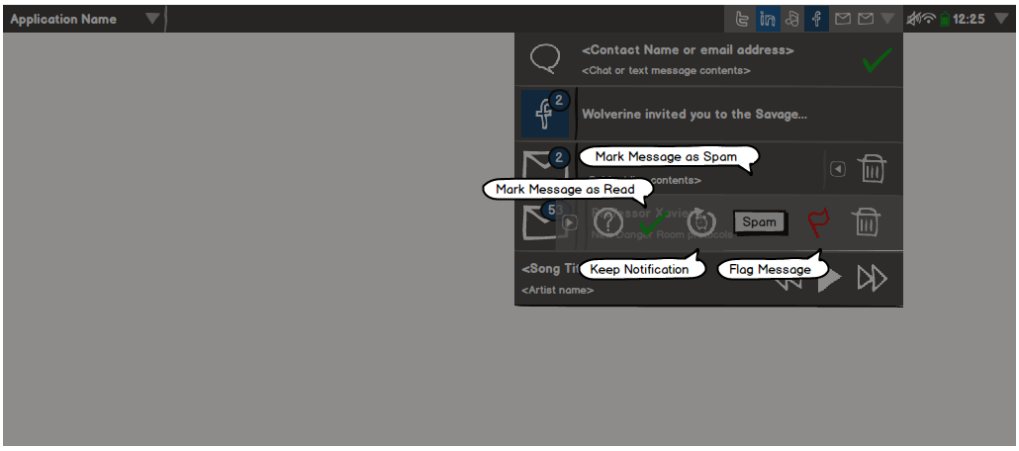
2) After swiping down on the notification area



3) After expanding the action drawer, drawer expanded



4) InfoTips invoked via tapping the Help icon



Improving Notifications v0.1, webOS Style

About v0 vs. v0.1

Whichever of the previous alternatives for the command drawer turns out to be best, they all rely mostly on icons, even if I choose to allow buttons. This brings up two issues:

1) If there are no buttons, the meaning of icons may not be clear. How do we make that clear?
2) If there **are** buttons, horizontal space will be problematic for label strings that require wider buttons. This is a problem whether the command drawer is horizontal or vertical/next to the notifications frame, but it is worse for the vertical version. This suggests the choice to restrict the command drawer to containing just icons.

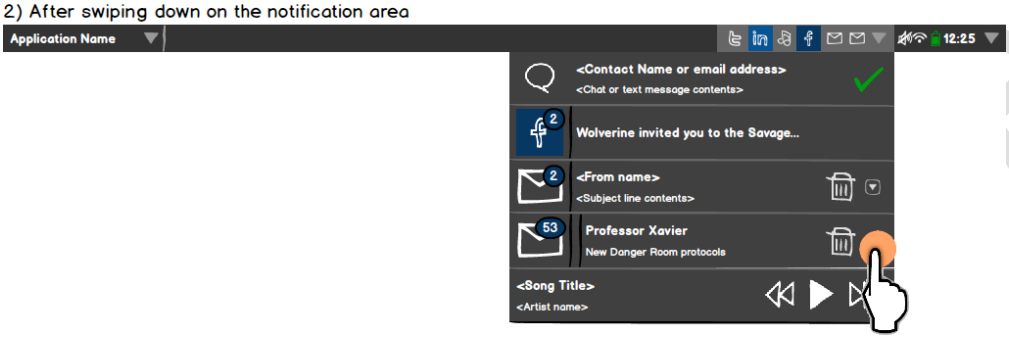
Between these two issues, one solution that presents itself is restricting the command drawer to icons, but adding in a standard icon to invoke "help" in the form of simultaneous "tooltips", such that small, left-aligned text labels appear over each icon. Those labels could allow for multiple lines of text, or just really long labels.

Design v0.1b: Action drawer opens below the notification

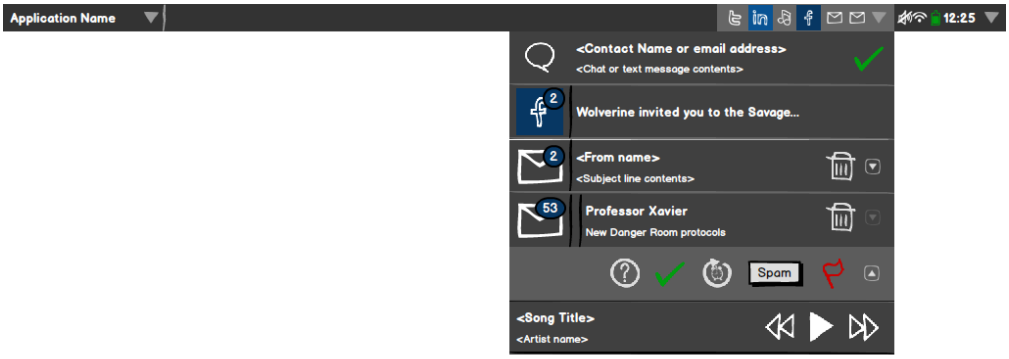
1) Status bar from within an app.



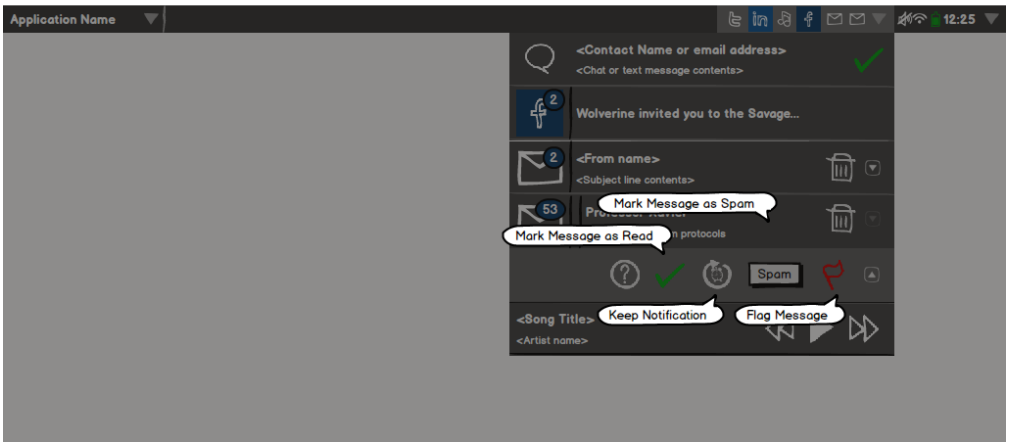
2) After swiping down on the notification area



3) After expanding the action drawer, drawer expanded



4) InfoTips invoked via tapping the Help icon



Improving Notifications v0.1, webOS Style

About v0 vs. v0.1

Whichever of the previous alternatives for the command drawer turns out to be best, they all rely mostly on icons, even if I choose to allow buttons. This brings up two issues:

1) If there are no buttons, the meaning of icons may not be clear. How do we make that clear?
2) If there **are** buttons, horizontal space will be problematic for label strings that require wider buttons. This is a problem whether the command drawer is horizontal or vertical/next to the notifications frame, but it is worse for the vertical version. This suggests the choice to restrict the command drawer to containing just icons.

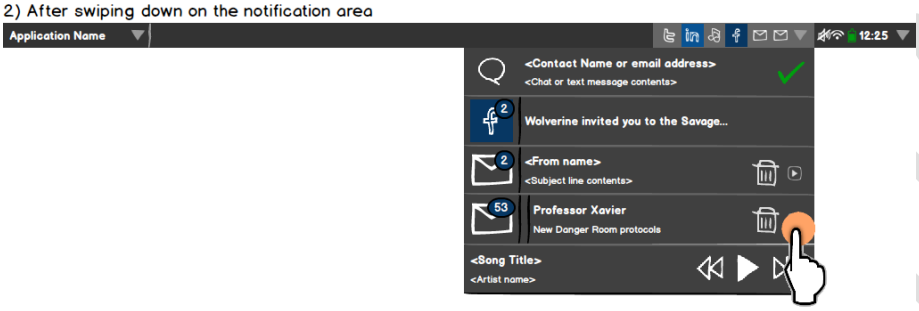
Between these two issues, one solution that presents itself is restricting the command drawer to icons, but adding in a standard icon to invoke "help" in the form of simultaneous "tooltips", such that small, left-aligned text labels appear over each icon. Those labels could allow for multiple lines of text, or just really long labels.

Design v0.1c: Action drawer opens below the notification

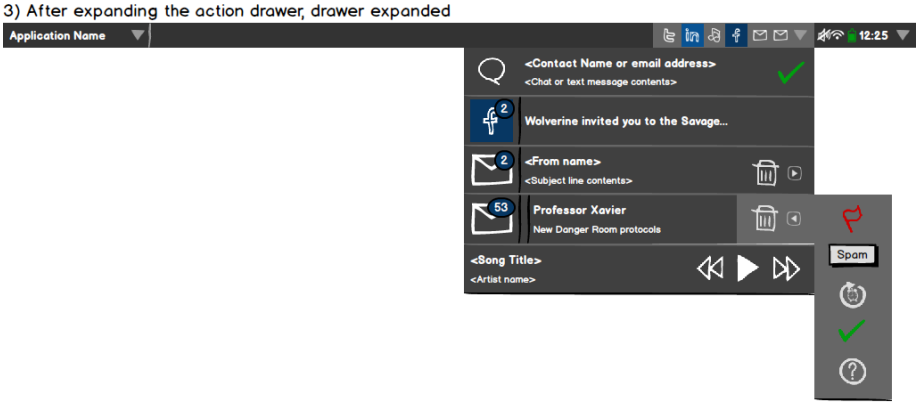
1) Status bar from within an app.



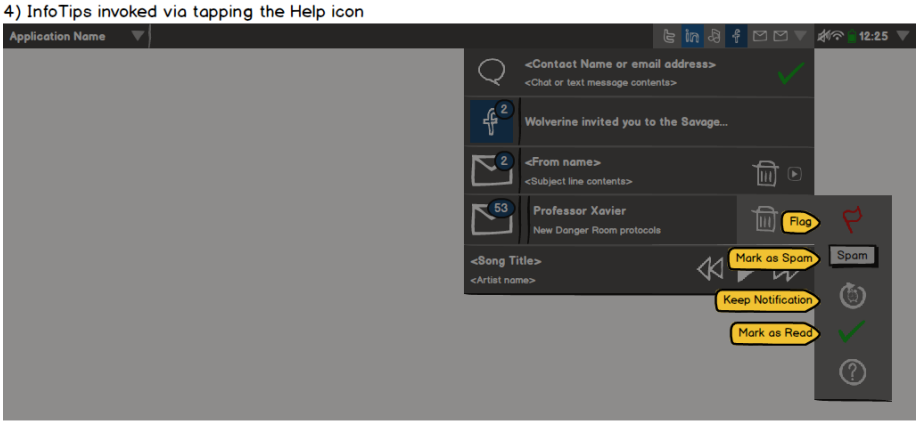
2) After swiping down on the notification area



3) After expanding the action drawer, drawer expanded



4) InfoTips invoked via tapping the Help icon



The fact that these InfoTips have yellow backgrounds where the other ones are white balloons is an artifact of my wanting pointy button shape, not to reflect any other difference. the pointy button shape didn't allow any change in color.