Procrastinate: How I Learned to Stop Worrying About Email and Love Procrastinating

Benjamin V. Hanrahan
Department of Computer Science
Virginia Tech
bhanraha@vt.edu

Manuel A. Pérez-Quiñones
Department of Computer Science
Virginia Tech
perez@cs.vt.edu

ABSTRACT
Throughout the course of a typical day, users send and receive a large number of emails. These emails are not created equal however, some can be dealt with immediately, while others cannot be completed in a single session. Existing email tools are well suited for the most common types of tasks, however, there is poor support for assisting in other types of email tasks. For example, emails that require effort or time to process are often postponed by users for later processing. Dabbish found that 37% of emails that require a reply are deferred to a later date. Current email tools do not support this case. The user is left to create ad-hoc solutions to prospectively remember the pending email task. Instead of leaving users to create their own ad-hoc method for prospectively remembering to handle email tasks, there should be direct support in the email client for this common activity. In this paper, we present *Procrastinate*, an additional email functionality built for Gmail that allows users to postpone working on an email to a specified future date. Using this functionality, users tag and archive emails with a future date, such as 11/25, or a relative one, such as "tomorrow," or "monday."

Author Keywords
Email, PIM, Prospective Memory

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous—*Optional sub-category*

General Terms
Design

INTRODUCTION
Throughout the course of a typical day, users send and receive a large number of emails. As of 2006, the average was 87 emails a day [3]. These emails are not created equal however, some can be dealt with immediately, while others cannot be completed in a single session [6]. Existing email tools are well suited for the most common types of tasks: crafting a quick reply to the author; deleting an uninteresting email; or even filing an email away.

However, there is poor support for assisting in other types of email tasks. For example, emails that require effort or time to process are often postponed by users for later processing. Dabbish found that 37% of emails that require a reply are deferred to a later date [2]. Current email tools do not support this case. The user is left to create ad-hoc solutions to prospectively remember the pending email task. Some of these solutions typically include: leaving the email in the inbox, increasing the amount of distractions in the inbox and increasing the cognitive load for processing email; file the email away and hope to remember to handle it at a later time; or offload the postponed task to another application such as a todo manager, to be revisited later.

However, even relatively simple solutions are abandoned, Whitaker reported that of the users that attempted to use a todo folder for pending emails, 95% abandoned the practice due to the additional cognitive step [7]. This result points to the probable abandonment of filing into a todo manager, as there are additional, more expensive cognitive steps involved.

Instead of leaving users to create their own ad-hoc method for prospectively remembering to handle email tasks, there should be direct support in the email client for this common activity. In this paper, we present *Procrastinate*, an additional email functionality built for Gmail that allows users to postpone working on an email to a specified future date. Using this functionality, users tag and archive emails with a future date, such as 11/25, or a relative one, such as "tomorrow," or "monday." Once the future date arrives, the email is moved back into the inbox and marked as unread. Using this functionality users are able to reduce the cognitive burden of prospective remembering to handle the task and the effort associated with refinding the email. Instead of the user refinding the email, *Procrastinate* makes the email *find the user* by moving the email back into the inbox at the appropriate time.

Instead of building a separate tool for this functionality, we opted to integrate it with email, following the spirit of Jones' Planz [4] and Bellotti's TaskMaster [1]. In effect, we propose to add simple todo functions to email clients with minimal impact to the interface.

RELATED WORK
There has been a relatively large amount of research in studying the mechanics of prospective memory. We draw on the work of Kliegel et al. and their outlining of the prospec-

Submitted for review to PIM 2012.

tive memory mechanism. The four phases of the prospective memory mechanism [5], as related to the email task with which we are concerned:

1. Intention formation - this is the point of realization that the user is going to have to handle this email later
2. Intention retention - this is the period between planning to handle the email and actually handling the email
3. Intention initiation - when the user actually starts the task
4. Intention execution - actually handling the email

In this project we aimed to reduce the cost associated with the formation, retention, and initiation stages. These stages represent points in the process where the email has to be initially handled in some way (formation), the email must be kept in memory (retention), and finally, the email must be located to be processed (initiation).

These different stages were recognized under a different name by Whittaker and Bellotti. "Task management is complicated because users have to remember to execute tasks..." relates to the Intention retention phase, while "Task management related to refinding activities" relates to the Intention initiation phase [6].

Research into email usage has established that users are appropriating email clients for task management as well. Whittaker described several types of emails which had pending actions associated with them [7]. Bellotti and Ducheneaut found that workers spend most of their time in email, often performing task management [1]. Dabbish also found that 35% of emails that require a response are being deferred by users [2].

As a result of their investigation into email Bellotti and Ducheneaut developed Taskmaster, which added significant amounts of todo functionality to email clients [1]. This project went a little further than what we are looking to do, we aim to add light-weight functionality that keeps email feeling like email.

POSTPONING ACTIONS ON EMAIL

For this project we wanted to add support for postponing emails with the least amount of impact on the user and email client. We felt that this common use case should not require an additional tool or any disruptions to the users' workflow. As such we wanted to implement it in such a way that the user was not required to install an add-on to their browser or email client, we also wanted to support this functionality in a way that it would be available on any device that had access to email. The use of IMAP makes this possible as manipulating an email is reflected on all devices connected to the account. Any platform of note has multiple clients by which to interact with email using IMAP as a communication of protocol.

This project started as a brainstorming session among members of the research group on Personal Information Management at Virginia Tech. We talked about scenarios where we felt that we were just spinning our wheels without accomplishing much. Several of us manage our work with a task

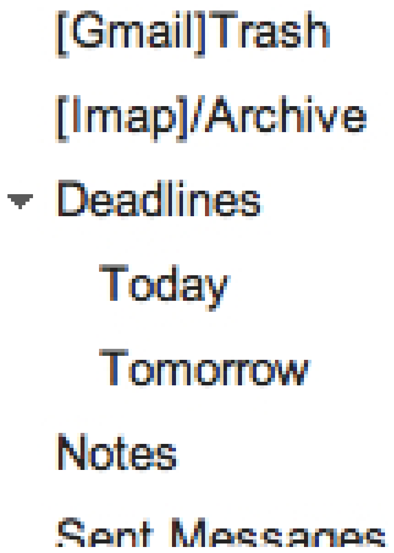

Figure 1. Gmail webclient, the deadlines tag has children tags with relative and absolute dates.

manager separate from the email program. In the conversation we realized that we all spend significant amount of time moving pending emails to a task manager and later looking over the task manager to handle the postponed email. The conversation lead to a "Wouldn't it be nice if we could postpone the email and have it show up in the inbox unread later on?" The result was *Procrastinate*. A quick check on the literature made us realize that we had an interesting implementation that addressed more than just our unique local needs.

Envisioned use

The primary scenario of use that we aim to support with this project is relatively simple, in this example scenario Linda is triaging her email inbox on Monday. As Linda is triaging her inbox she comes across an email from her boss that is requesting a status report on her project on Friday. Several important pieces of her project should be done by Thursday. She would like to write the status report on Friday morning, once she has received the update on the other pieces. She tags and archives the message with "Friday" which moves the email out of the inbox for the time being. The week goes by and on Thursday afternoon Linda has received the pieces of her project. On Friday morning, the original email from her boss appears in her inbox for processing as unread. Upon seeing it, she initiates execution on her handling of the email.

Implementation

To realize this vision we implemented the functionality via a Python script. To limit development effort we chose to support email accounts on Gmail only. Currently this script runs nightly via a cron job and accesses GMail accounts using OAuth which allows us to keep users' passwords safe (we don't store them in our implementation).

This cron job scans any tags (Gmail labels) that are listed under the "Deadlines" folder, a convention that we are using to

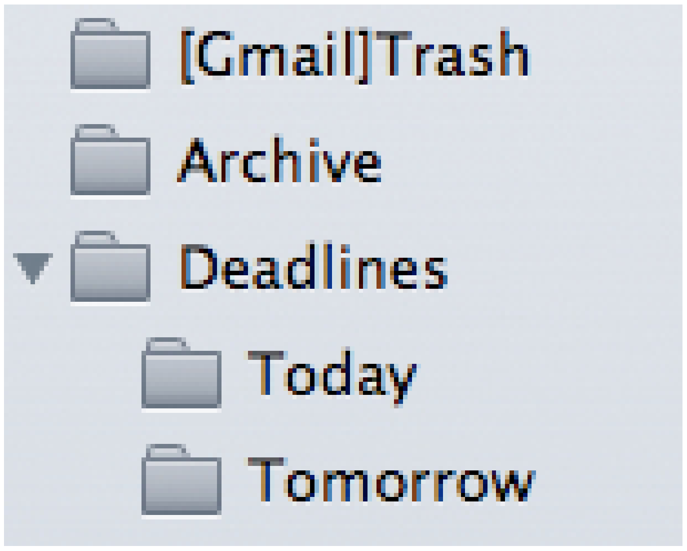

Figure 2. Gmail Mail.app, labels are interpreted as folders. The user simply moves the messages to the specified folder.

keep our application from operating on any unintended folders and keeping all relative date folders organized and away from the user's other folders. Under the "Deadlines" tag we scan for several tags. We process ones that have an absolute date (e.g., 11/15), ones that have a day of week (Friday), and messages with the tag "Tomorrow." All the messages found are moved from the existing folder to the inbox. In addition, these messages are marked as "Unread."

The use of the system is rather simplistic. The user applies these tags as s/he is reading email, and archives the message. This removes the postponed message from the inbox, and requires no effort on the user's part to get the messages back in the inbox. On the labelled date, the message will appear on the inbox as a new unread message.

Advantages of the approach

After several weeks of use, we have identified the following advantages of our approach:

1. It reduces cognitive load of having to remember (prospective remembering) tasks to do in the near future.
2. It reduces the time of processing the inbox, since tasks that require later processing are easily filed away and removed from the inbox as distractors. The other two options of how to process these types of messages are: a) leaving the message in the inbox. This increases the time to process the inbox by distracting the user and making him/her repeatedly processing the email that is postponed; or b) moving the email to a task manager (todo program), assigning a date or priority and routinely checking the task manager for pending tasks to be processed later. We feel our solution is better than either one of these as it requires less active processing from the user.
3. Our solution works well without any addition to the interface. All Gmail clients provide support for labels and archive, since both are such a central part of how this system works. In addition, the Gmail web client provides auto-completion for labels. In our test, we often type "tomorrow" and Gmail completes the label to Deadlines/Tomorrow.
4. Works on all devices. Email users today access their messages from multiple locations using multiple devices. In our experience, all devices provide some form of "label" or "archive" functionality that matches Gmail's use of labels. So, it is possible to read an email in your smart phone and 'file' it away (or *Procrastinate*) for action at a later date.

The current embodiment of this tool is deployed on our servers and under use by our lab members. So far we have experienced several positives from this new functionality. Of course, these are experiences within our research group, so it hardly constitutes a validation of our approach. Nevertheless, the initial experience (call it a beta-test) yielded the following insights:

- Much cleaner inbox.
- Less effort spent on managing emails and todos.
- We have all been surprised at how natural the tool felt

That said, we have also experienced some growing pains:

- Issues determining the frequency with which the cron job should run. Initially we had the cron job run in the individual's computers, so that we wouldn't have to deal with passwords. Unfortunately, we could not find a way to guarantee that the cron job would run at a time when the computer was awake. Because we use relative dates for filing messages (e.g. tomorrow), we couldn't afford to skip a day because the computer wasn't used that day. This often happened when we were on travel, the travel day we often used our phones for accessing emails, thus breaking the system. Since, we have moved to a server-based implementation that uses OAuth to connect to Gmail. That way we don't have to capture and store user's email in our system. They authenticate via OAuth and our system then runs at a predetermined time to process the labels.
- Where should the mails be put when they become relevant? The inbox may not be the correct place, however, if we use a folder that requires the user to navigate to check for todos this may lead to abandonment. We have found 2 distinct approaches to this from two of our users. One prefers the message to be physically moved to the inbox and labeled as unread. The other uses Gmail's priority inbox and has a section that shows the label "Deadline/Today." This user prefers to have the emails that reach actionable time simple be tagged with "Deadline/Today." We are sure that there are other strategies that our tool must support.
- Adding new users always takes a few minutes of training as the functionality is not intuitively afforded.
- We found that we wanted to add reminders that did not originate on email. For example, adding a reminder like "Complete the PIM paper" and tagging it with 11/25 knowing that the message will be among the unread emails in the

morning of the 25th. There is no easy way to do this with our current implementation. But more importantly, we are not convinced that this is one of the use cases that we want to support. We need to explore with real users to see what their opinion of this feature is.

- We also discovered a second scenario of use. Because Gmail saves conversation threads together under the label applied to any of its individual messages, PostPone brings back not just individual messages but full threads. So, we found that at times we send an email requesting some information. One of our users found that it was appropriate to set a "reminder" for a few days later to see if a reply to the message had come back or not. He tagged sent messages with a day of the week that was 2 days later, in effect using *Procrastinate* to help check if someone has replied to the original message. When the email comes back to the inbox, it comes back with the full thread, so if a response has come back, the user sees it and can quickly archive the thread. If there has been no response, then the user can send a reminder and *Procrastinate* again.

FUTURE WORK

Overall, we are very satisfied with the simplicity of the approach and the fact that it requires little adaptation from our part. The immediate future plans for this project are to deploy it to a larger group of people and gather data about how people use it. Some of the research questions that we are interested in exploring include:

1. How do people use *Procrastinate* for email/task management?
2. What types of strategies evolve out of using this tool?
3. How many emails are postponed for later processing? Are messages postponed multiple times? Is there a day of the week when people handle most of their postponed work? (that is, do people schedule a marathon session for handling emails)
4. Should we create a lightweight plugin that will make *Procrastinate* easier, and help to reduce training?
5. Are there some user attributes that influence the use of *Procrastinate*? For example, we found users in our group that did not have that many emails, so they didn't postpone messages until later, simply handled them relatively soon after they saw it. Is the amount of email a reason why people postpone replying? Is it the complexity of knowledge worker's activities (e.g. number of different projects)?

In addition, we are interested in exploring how our tool can support collaborative work. For example, how hard would users find it to do some collaborative todos? Two users using *Procrastinate* can have a common set of shared tags that include each others name. Could Manuel assign work to Ben by tagging the message as "Assign/Ben" and having *Procrastinate* actually forward the note to Ben and mark it as "Pending" in Manuel's system for it to be checked later?

REFERENCES

1. V. Bellotti, N. Ducheneaut, M. Howard, and I. Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 345–352, New York, NY, USA, 2003. ACM.
2. L. A. Dabbish, R. E. Kraut, S. Fussell, and S. Kiesler. Understanding email use: predicting action on a message. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 691–700, New York, NY, USA, 2005. ACM.
3. D. Fisher, A. J. Brush, E. Gleave, and M. A. Smith. Revisiting Whittaker & Sidner's "Email Overload" ten years later. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 309–312, New York, NY, USA, 2006. ACM.
4. W. Jones, D. Hou, B. D. Sethanandha, S. Bi, and J. Gemmell. Planz to put our digital information in its place. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, CHI EA '10, pages 2803–2812, New York, NY, USA, 2010. ACM.
5. C. Roda. Human attention and its implications for human-computer interaction. In C. Roda, editor, *Human Attention in Digital Environments*, pages 11–62. Cambridge University Press, 2011.
6. S. Whittaker, V. Bellotti, and J. Gwizdka. Everything through email. In W. Jones and J. Teevan, editors, *Personal Information Management*, chapter 10. University of Washington Press, 2007.
7. S. Whittaker and C. Sidner. Email overload: exploring personal information management of email. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 276–283, New York, NY, USA, 1996. ACM Press.