# Authoring Email Automation Rules

**Soya Park**
soya@mit.edu
MIT CSAIL

**Amy X. Zhang**
axz@mit.edu
MIT CSAIL

**Luke S. Murray**
lsmurray@mit.edu
MIT CSAIL

**David R. Karger**
karger@mit.edu
MIT CSAIL

## ABSTRACT

Email users perform repetitive, automatable tasks over a large volume of email daily. We conducted a range of need-finding studies on automating email management and identified general categories of email automation needs and opportunities. Here, we provide a summary of our previous work and future directions for our work.

## CCS CONCEPTS

• **Social and professional topics** → **Automation**; • **Information systems** → *Email*.

## KEYWORDS

email; task management; perosnal information repository

**ACM Reference Format:**
Soya Park, Amy X. Zhang, Luke S. Murray, and David R. Karger. 2019. Authoring Email Automation Rules. In *CHI '19: SIGCHI Conference on Human Factors in Computing System, May 04–09, 2019, Glasgow, UK*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

ACM ISBN 978-1-4503-9999-9/18/06...$15.00
https://doi.org/10.1145/1122445.1122456

## INTRODUCTION

Email has emerged as a central medium for everyday tasks and is used extensively by knowledge workers. Email users conduct tasks such as scheduling and information exchange on a daily basis. These tasks are numerous, so it is hard for users to track and manage their attention manually. For instance, there are many emails involving deadlines (e.g., reply to an application by next Wednesday), but most current email tools do not help users keep up with these deadlines. Users may be able to add a deadline label to the message but the email interfaces does not provide sorting by dates in these labels or reminders of approaching deadlines. A user may create a reminder in their calendar, but the disconnection of email from calendar means a user has to waste time searching for it when reminded. This does not take long time per a individual message, but over many emails adds up to significant wasted time.

In previous work [8] we studied how users would like to automate their email workflows. We conducted three different research probes. Our main observations were threefold:

"*If the email has a deadline on it, label the email with the date of the deadline so that later I can sort by deadline.*"

"*Notify me immediately when an urgent email comes, or an email from an important person (e.g my boss), otherwise only notify me at set periods of the day (e.g. twice a day) if there are other emails*"

"*Move important emails to the top in a certain time in the day, maybe around 1pm. Or at least remind me to open them.*"

- Richer data model: Many attributes that users would like to rely on for automation are not part of the existing email data model. Unlike the current email message data structure which only contains invariant attributes such as sender, subject line, email users want to leverage additional, mutable information such as progress, deadline, topic, priority, and task to manage their messages. They also want to annotate their contacts with various attributes to *query* their contacts by their relationship, organization, geo-location, etc.
- Leveraging users' context: Users want their email to behave differently based on current *contexts*. We identified two different types of context: *internal* context is a state of their inboxes, such as how many messages are unread, and *external* context is users' real-world situation such as their current activity and geo-location. Our participants suggest different email automation ideas incorporating their context.
- Attention management: The ultimate goal of the needs mentioned above is to draw users' attention to the right message. The participants shared their heuristics of triaging emails based on the attributes of each message and their strategy to bring their attention to important messages. In addition, users wish to author rules for push notification of a message arrival and alter visualization of their email interface to highlight important (in current context) emails.

These finding show that email users have specific, well articulated ideas for email automation. But at present there is no way for users to *tell* the system what they want. To this end, we propose to gain insight from the field of end-user programming to provide email users with a tool where they can *express* their automation rules without programming skills.

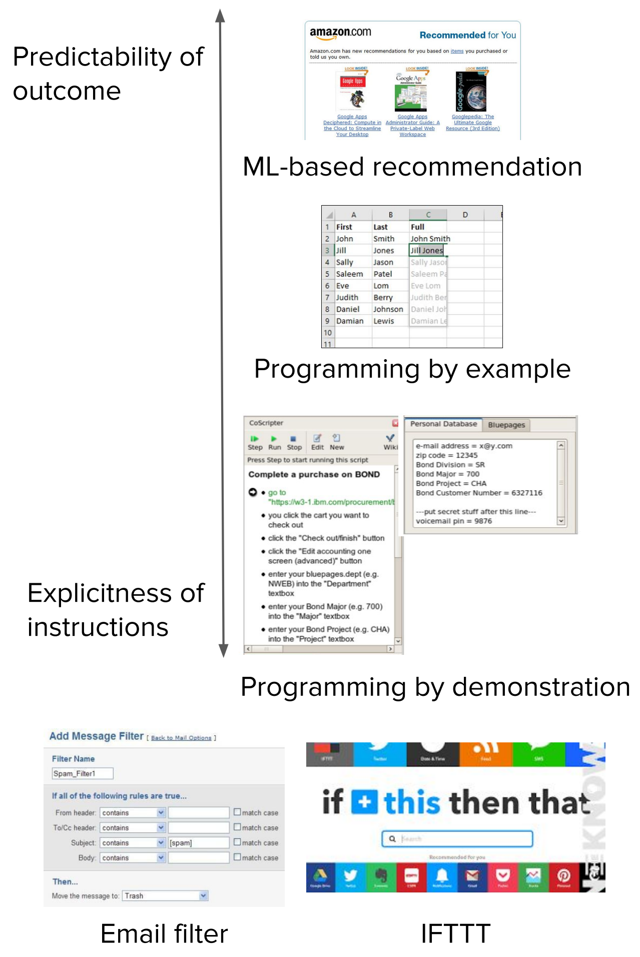
Predictability of outcome
ML-based recommendation
Programming by example
Programming by demonstration
Explicitness of instructions
Email filter
IFTTT
if this then that

**Figure 1: Spectrum of task automation**

## RELATED WORK

Given the repetitive nature of everyday tasks, automation tools for knowledge workers have been explored through many different lenses by the HCI community and deployed in popular applications. Scholars have proposed different techniques to support automation as shown in Figure 1. One is *inferring* intended automations by learning from massive user activity logs, while another is letting users *explicitly articulate* what they want. Both possess interesting pros and cons. Predicting automations does not require users’ input, evolving as the system collects more data. However, such machine learning approaches require a large quantity of data for a system to learn and it is difficult to customize the resulting automations. Explicit instruction gives users more control to specify and customize their automations. However, these instructions must be imagined and communicated by the user. Here, we discuss current approaches to task automation of different degrees of explicitness in the realm of end-user programming, then focus on email automation.

### Automation for Knowledge Workers

Many applications used by knowledge workers offer automation interfaces that let users *program* automations. Since many knowledge workers can’t code, those applications must enable their users to programming *without coding*.

Microsoft Excel automatically detects data formatting after a user provides a few examples. Under the hood, the system tries to synthesize the intended data formatting based on the examples [3, 5]. The system can then automate formatting of other cells as data is entered. This notion of end-user programming is called *programming by example* (PBE). Users specify the program by providing pairs of input (here, data) and correspondiing output (here, formats). Alternatively, there are *programming by demonstration (PBD)* systems where users show a series of actions to teach the system their intent to generate a program [6]. These approaches help users to generate programs easily without coding by users. But after a program is generated, there are still remaining tasks – debugging and reusing the program. This is difficult for users who do not know how to code.

Instead of generating a program in a programming language, an alternative is to define programs in *controlled natural language (CNL)*. This approach permits users to write programs in a restricted, unambiguous subset of natural language sentences. For instance, Atomate [10] provides an interface where a user builds a sentence incrementally by using a dropdown menu to select among permitted verbs (tests and actions), nouns (entities), and adjectives (entity attributes).

### Programming Email Rules

Most email interfaces provide email filters. Users can author rules that are applied to incoming emails. Email filters correspond to one of explicit end-user programming interface. Users can articulate a set

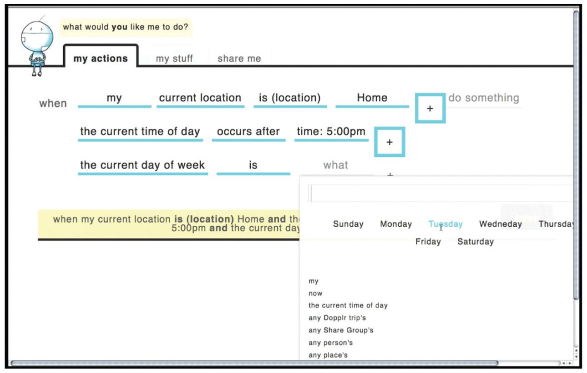

**Figure 2: Atomate[10] lets users automate their workflow without programming skills. It generates automtation programs in controlled natural language (CNL).**

of rules: rules are consists of two parts triggers and actions. If there is incoming email that triggers (satisfy) user-defined conditions (e.g., from certain senders, contains keywords), it performs defined actions on the email. Advanced email users can host their own email engine on their own server and program their email rules in code [2, 9]. Beyond managing incoming emails, there are tools that remind users to check emails at scheduled time [1] and automate workflow with 3rd party applications [4, 7]. All these tools provide straightforward interfaces to users to articulate their automations explicitly.

## OUR WORK

The background above discusses two aspects of email automation: a suitable *model* rich enough for the rules the user wants to specify, and a *language or interface* that lets author rules in the model. Our previous work elicited insights about the necessary model from non-coding email users. As a further step, we developed an environment in which *coders* could program against this richer model using python. **YouPS** connects to a user's IMAP server and exposes it to the programmer as a simple python environment with objects for each email, thread, folder and contact and methods for manipulating them. It also provides an editing environment for creating small programs over this model. Our preliminary deployment of YouPS yielded evidence that our extended email vocabulary is suitable for scripting the rules users want.

We aim to continue to extend our model to capture more of what users need to create the rules they want, for example with richer functionalities such as executing timer events and connecting with users' calendars.

### Email Automation for Non-programmers

In parallel with our extension of the data model, we aim to draw on the ideas of end-user programming to extend our email automation capabilities to non-programmers by creating non-coding interfaces for expressing rules. For initial guidance we can look to existing email clients' interfaces that permit non-programmers to create filters; these interfaces generally consist of a simple trigger (condition) and action. They offer drop-down menus of attributes and constraints that can be applied in filtering. We believe such an interface could be augmented with the new concepts identified in the work presented previously. We will also consider alternatives, e.g., block programming but with support for reusing and remixing email rules, in a way that is usable for non-programmers.

## REFERENCES

[1] Boomerang. 2016. boomerang. http://www.boomeranggmail.com.
[2] Philip A. Guenther. 1990. procmail. https://linux.die.net/man/1/procmail.
[3] Sumit Gulwani. 2011. Automating String Processing in Spreadsheets Using Input-output Examples. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '11).* ACM, New York, NY,

USA, 317–330. https://doi.org/10.1145/1926385.1926423
[4] IFTTT. 2010. IFTTT Gmail. https://ifttt.com/gmail.
[5] Vu Le and Sumit Gulwani. 2014. FlashExtract: A Framework for Data Extraction by Examples. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '14)*. ACM, New York, NY, USA, 542–553. https://doi.org/10.1145/2594291.2594333
[6] Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1719–1728. http://doi.acm.org/10.1145/1357054.1357323
[7] Microsoft. 2018. Microsoft Flow. https://flow.microsoft.com/en-us/.
[8] Soya Park, Amy X. Zhang, Luke S. Murray, and David R. Karger. 2019. Opportunities for Automating Email Processing: A Need- Finding Study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing System (CHI '19)*. ACM, New York, NY, USA, 12. https://doi.org/10.1145/3290605.3300604 . To appear.
[9] CMU Cyrus Project. 2008. sieve. http://sieve.info.
[10] Max Van Kleek, Brennan Moore, David R. Karger, Paul André, and m.c. schraefel. 2010. Atomate It! End-user Context-sensitive Automation Using Heterogeneous Information Sources on the Web. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 951–960. https://doi.org/10.1145/1772690.1772787