Personalized Experiences for End-User Programming on the Web

Max Van Kleek
CSAIL, MIT
Cambridge, MA, 02319, USA
max@mit.edu

Paul André
ECS, University of Southampton
SO17 1BJ, UK
pa2@ecs.soton.ac.uk

David Karger
CSAIL, MIT
Cambridge, MA, 02319, USA
karger@mit.edu

m.c. schraefel
ECS, University of Southampton
SO17 1BJ, UK
mc@ecs.soton.ac.uk

Abstract
In this position paper we explore current work in AtomsMasher, an end-user reactive programming environment for the Web, highlight ongoing work in user interface design, privacy, and sharing, and look towards a future of extending end-user programming from simple tasks to complete experiences.

Introduction
In the past several years, the ubiquity of RSS feeds (published summaries of updated content) has grown beyond public web sites and blogs. Web2.0 sites have encouraged publishing of a wealth of personal, social and contextual information: facebook actions, twittering, calendar appointments, music interests, location, etc. These data sources are of some value individually, for personal consumption. We hypothesize that these sources could serve a greater purpose: driving simple reactive tasks for personal information needs. By this, we mean easy ways to combine and filter sources that, given a certain state, trigger an action. For instance, when I walk past a concert hall, look at mine and my nearby friends' music interests, and alert me if there is a concert soon we may enjoy.

Existing work in mashups enables blending and filtering for unique views on information, but provides no active capabilities. Active scripting in end-user programming is so far limited to closed homogenous models, not allowing scripting over heterogeneous sources, or personal context from desktop activity. In our approach, we identify three core challenges that need to be addressed to deliver effective reactive behaviors that will let us do more with our information with less effort. These challenges are: a) interrogating whether current data feeds have sufficient, timely information to drive useful reactivity; b) finding a flexible and scalable method for consolidating and integrating heterogeneous information from diverse web sources; c) designing a user interface for end-users that can make the specification of desired automation easy and error-free.

The result of addressing these challenges is AtomsMasher, an end-user reactive programming application/rule based system, driven by web data. AtomsMasher presents a constrained simplified natural language (NL) UI to allow users to specify actions they want done and the conditions under which these actions should be done. It then determines when these conditions are met based on a rich internal RDF world model it has consolidated out of web data feeds and desktop activity observations. In the following sections we elaborate on the components of AtomsMasher, highlight ongoing work in user interface design, privacy and sharing, and conclude with a look at the future of end-user programming and book chapter suggestions.

Capturing Data

The availability of even a little bit of structured information that we can capture and associate automatically means we enable many new ways to interrogate and build behaviors over our information. We have begun to explore context capture possibilities [9] in order to provide contexts for reuse and rediscovery. In addition to the social and public data available, we have a framework to obtain personal structured information directly from the users' computer [7]. In the next section, we discuss how the blending of these sources of information can automate many of our current manual information management tasks.

Towards Reactive Behaviors

In this section we briefly discuss the core components of AtomsMasher: rules to create reactive behaviors, the rule creation user interface and simulation sandbox, and a quick look behind the scenes. (A previous description of AtomsMasher is in [8], with a more up-to-date full paper in submission to WWW).

Rules in AtomsMasher

Rules constitute the basic unit of AtomsMasher's reactive behaviors. Like other rule-based systems, each of AM's rules consist of a set of conditions that characterise the situation under which a rule should be executed (known as the *antecedent* or *if-part*), and the set of actions to be carried out when these conditions are met (known as the *consequent*, or *then-part*). The conditions in the antecedent are expressed as simple conjunctions of predicates, each of which may take entities or values as arguments. Rules in AM consist of one of two types: those that update AM's internal world model based on new events arriving externally, and those that take external action based on updates to the internal model. Keeping rules separated into these types decouples knowledge sources from action, making it easier to scale AM to new sources and actions.

Simplified Natural Language Interface

Previous efforts [6] have discovered several challenges to having end-users specify rules in situations where they desired some context-aware automation. Central to our approach with AM is a guided input simplified natural language interface that lets users express rules in a form resembling natural English (inspired by [2][3]). We apply these ideas to behavior specification, which requires users to specify the conditions when they want something done (the rule antecedent, previously described), and the action(s) they want done (consequent) when those conditions are satisfied - these expressions are built incrementally through a sequence of entity, predicate, actions and value selectors (each an auto-completing drop-down box) for specifying components of clauses and actions (fig. 1). We add several additional features to make this more natural, including demonstrative pronouns, and a rule simulator inspired by PigPen [5] to catch erroneous specification of rule components immediately after they are created.

Architecture and Data Model

Space constraints do not allow for an in-depth discussion, but one key issue is the use of a single persistent internal representation containing simple key representations of people, places, events and resources. This served three roles: a single representation avoids the pairwise-alignment problem that serves as a scalability problem to many mashups; a single unambiguous world model for the AM rule chainer; and finally this representation serves as an important abstraction barrier, allowing information sources to be exchanged freely (or added for redundancy) without having to modify user behaviors.

User Interface Design

While end-user programming for the web has enjoyed some success, we believe the lack of truly end-user friendly interfaces is a significant barrier to adoption. Exciting recent work in Yahoo! Pipes still "*needs a lot more work on the user interface to make it easier to use*"[1], and a user study of the EUP tool Marmite found it was only easy to use for "*programmers and spreadsheet users*" [10]. While there has been previous work in visual programming metaphors and programming-by-demonstration, there is still significant work to be done to simplify the initiation, understanding and completion of actions, and scrutability of behaviors. In the meantime, we hope our user interface (described previously) and the design and evaluation of the AtomStasher (next) could help towards this goal.

Extensibility and Sharing

In AtomsMasher, we encourage the re-usability and sharing of behaviors by shielding variables from direct access to the information sources, preventing authors from writing their behavior specific to a particular source, and allowing the system to scale to new data sources. As we have elaborated elsewhere [1], the social community data that inspired AM is part of a wider social evolution on the Web. By establishing the "AtomStasher" (similar to the Co-Scripter wiki [4]), we aim to make actions shareable, encouraging an active community, and allowing less experienced users to download more complex rules that others have written.

Privacy

With users publishing increasing amounts of personal information and interactions on the Web, and user interfaces doing little to alert users to options or the potential dangers, privacy is a significant area for future work. In AtomsMasher, there are obvious concerns in blending personal and Web data, though by running AM client-side we hope to retain control over any potential problems. In a broader sense, AM may create its own privacy implications. By increasing the ease of

[1] http://radar.oreilly.com/archives/2007/02/pipes-and-filte.html

combining multiple sources of data about a friend, AM highlights how much personal information is being broadcast to the Web, and enables inference and reactive behaviors based on that information. It remains to be studied what users' major privacy concerns regarding AM are.

Future of End-User Programming

Beyond completion of specific actions, we see a wider future for end-user programming in which a user's entire Web experience is customized. Take search, for instance, one of the most popular activities on the Web. Live Search macros[2] allow creation of "your own search engine", tailored to a specific topic. The search engine Viewzi[3] provides user-defined views onto specific data sources, a cookbook user interface for searches within recipe sites, for instance. These examples indicate the potential for a completely customizable Web experience, and it remains to be seen how an entire suite of applications with a truly end-user interface could provide such an experience.

Book Chapter Suggestion

Dependent on the space and scope of the book chapter, we would like to elaborate on our general approach to end-user programmable definitions of 'reactive behaviors', our vision of how personal information management can benefit from reactive behaviors driven by web data, and results from our first deployment of AtomsMasher. There are three key attributes to our approach: easy data alignment, easy definition of behaviors, and availability of context information to support these. Overall, there is a compelling user interaction challenge to be addressed. As examples of how each of these components may be realized, we would present our thoughts on data unification and RDF, our Personal Lifetime User Modeling (PLUM) framework, and how the context it provides allows the unique reactive behaviors seen in AM. A validation of our user interface and general approach would ideally be provided by a deployment we plan for Spring 2009, as well as allow reporting on the innovative uses we hope to observe.

[2] http://search.live.com/macros
[3] http://www.viewzi.com

Acknowledgements

Our previous and ongoing work in this area has been developed in conjunction with Michael Bernstein and Rob Miller (MIT), Daniel Smith and Max Wilson (Southampton), Mikko Perttunen (Oulu), and Ora Lassila, Mark Adler and Deepali Khushraj (Nokia Research Center Cambridge).

References

[1] André, P., schraefel, m.c., Wilson, M.L. & Smith, D.A. *The Metadata is the Message*. Web Sci Workshop at WWW 2008.

[2] Bernstein, A. & Kaufman, E. *GINO - A Guided Input Natural Language Ontology Editor*. ISWC2006.

[3] Kaiser, C. *Ginseng - A Natural Language User Interface for Semantic Web Search*. Thesis, University of Zurich.

[4] Leshed, G., Haber, E. M., Matthews, T., and Lau, T. 2008. *CoScripter: automating & sharing how-to knowledge in the enterprise*. CHI 2008.

[5] Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins A., *Pig Latin: A not-so-foreign language for data processing*. SIGMOD 2008.

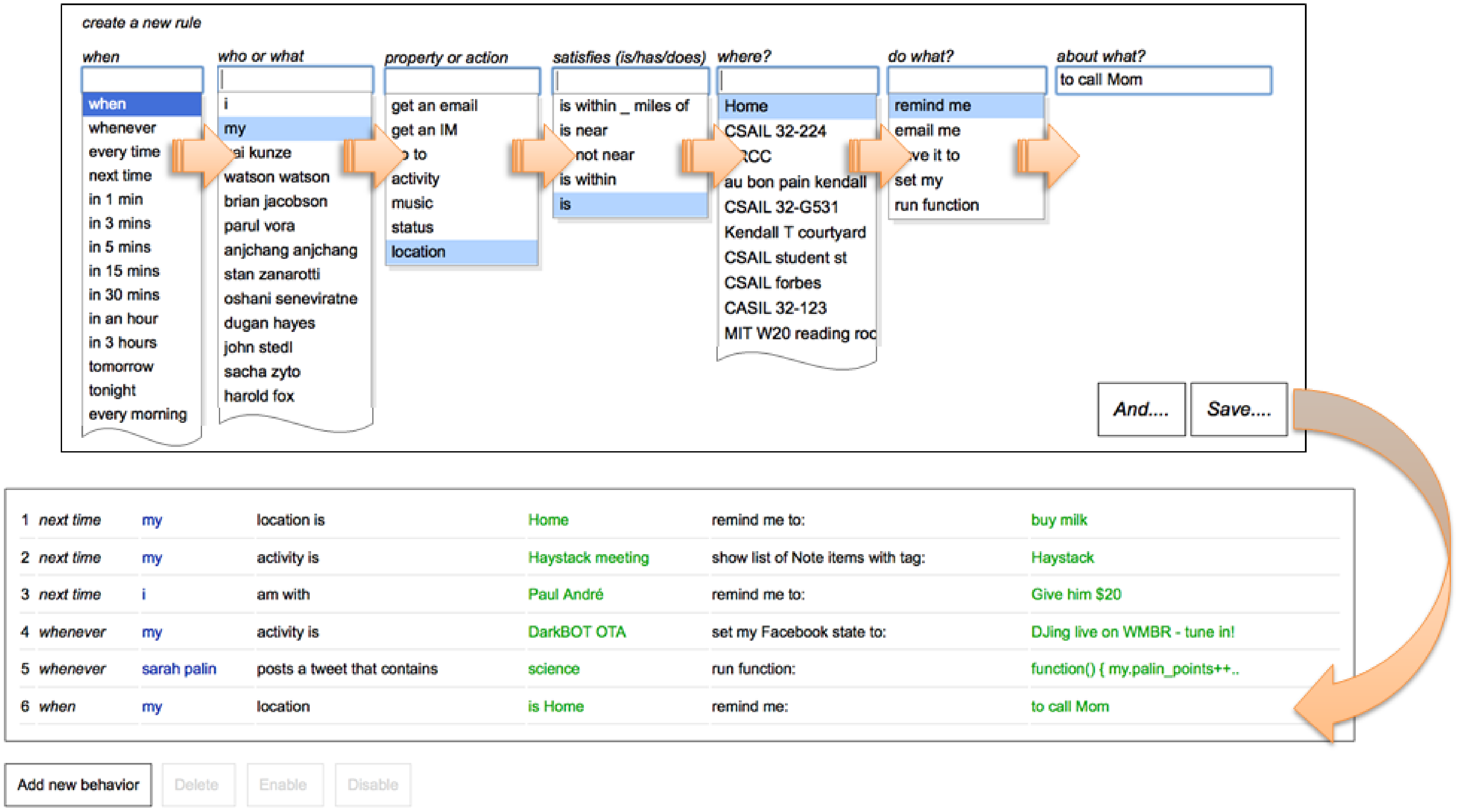
[6] Sohn, T. and Dey, A. 2003. *iCAP: an informal tool for interactive prototyping of context-aware applications*. CHI 2003.

[7] Van Kleek, M. & Shrobe, H. *A Practical Activity Capture Framework for Personal Lifetime User Modeling*. UM2007.

[8] Van Kleek, M., André, P., Perttunen, M., Bernstein, M., Karger, D., Miller, R. and schraefel, m.c. *AtomsMasher: Personal Reactive Automation for the Web.* UIST2008

[9] Van Kleek, M., Bernstein, M., Karger, D. & schraefel, m.c. *Gui – Phooey! The Case for Text Input*. UIST 2007.

[10] Wong, J. & Hong, J.I. *Making mashups with Marmite: Towards End-User Programming for the Web*. CHI 2007.



about what? to call Mom

1	*next time*	my	location is	Home	remind me to:	buy milk
2	*next time*	my	activity is	Haystack meeting	show list of Note items with tag:	Haystack
3	*next time*	i	am with	Paul André	remind me to:	Give him $20
4	*whenever*	my	activity is	DarkBOT OTA	set my Facebook state to:	DJing live on WMBR - tune in!
5	*whenever*	sarah palin	posts a tweet that contains	science	run function:	function() { my.palin_points++..
6	*when*	my	location	is Home	remind me:	to call Mom

Figure 1. A user creating a reminder rule using AtomsMasher's constrained-input simplified-natural-language (CSNLI) UI (sequence shown for brevity). The user first specifies when the rule should run (antecedent), then what it should do (the consequent). Input boxes restrict their values to only valid inputs given the values provided so far.