Re-forming the Internet with its End Users

Michael Toomim
University of Washington

1. Introduction

Describe real world setting

Identify problem

Almost a quarter of the world's population—1.4 billion people—now use the Internet, and are thus tapping into mankind's most exciting technology for communication, collaboration, organization, and mass coordination of behavior towards shared goals. However, the websites that coordinate and broadcast this information are controlled and created by a relatively small cabal of programmers—there are only a few million software engineers in the world.

Although we hope these programmers solve the problems that end users face, they can only be paid to if there is a clear business case to be made for a profitable market of users. Even then, end users are at the mercy of business goals, since the business owner decides how to modify and extend the website, adding or improving features that affect the lives of the end users. The programmer for my online banking website implemented an HTML table that displays my transaction history, but not a simple graph or chart of the information that would help me manage my finances. My phone service website records and displays my phone call behavior, and internally mines this data for business calculations of how much revenue they extract against my phone usage, but the website does not surface this analysis to me to suggest a cheaper plan that saves me money.

Describe approach to solve the stated problem

I envision an Internet where anybody can patch any website to support features they want. An Internet where an end user with an idea need not convince Facebook or Bank of America to implement the solution to their need. Where Internet shoppers improve their own shopping user experience, instead of relying on the adversarial whims of their e-merchants who can profit by withholding information. Where knowledge workers can create and improve the information systems that help them get their work done, rather than arguing with overburdened programmers. This Internet can be bootstrapped from our current one, using technology to overcome two technical barriers:

1. *Barrier of ownership*. Each website is controlled by a single entity, with its own motives, and small third parties have a disadvantaged position from which to influence their design.
2. *Barrier of programming*. Even if a user has access to the website's source, he still must have the programming skill to change the site, understanding HTML, PHP, relational databases, and SQL. This limits the number and types of people who can implement the changes that solve their domain-specific problems.

The following work attacks these two barriers.

2. Enhancing existing websites with new tasks

My work on the reform project [1] democratizes the ability to modify existing websites to support new tasks.

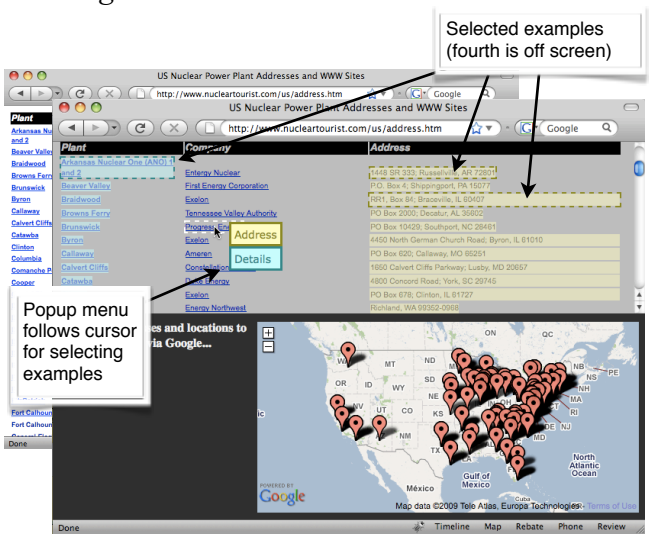

Figure 1: Before and after an end user made a map for a page of U.S. nuclear reactor locations. The process took five clicks.

Explain how this work is related to other existing work

Today, tools like Greasemonkey and Chickenfoot already provide the raw technology for modifying a website in the end user's web browser instead of the webmaster's web server by injecting it with a special browser-side script. The problem, however, is that it takes a programmer to enhance a website with such a script, and there are only a few million programmers—for too few to tackle the plethora of 175 million websites on the Internet.

reform instead leverages the Internet's 1.4 billion end users, allowing a single programmer to enhance many websites at once. A programmer authors a single site-independent web enhancement, and end users attach it to all the sites they use in the context of their existing tasks. This architecture of write-once apply-anywhere web enhancements divides web enhancement into two roles: programming and attaching. This allows end-users to do the attaching, and bring enhancements to many more sites.

The key is enabling end users to teach an enhancement how to attach to a new website and understand its data representation, a difficult problem traditionally studied as *web information extraction* or *web scraping*. reform presents a new interactive machine learning technique designed for

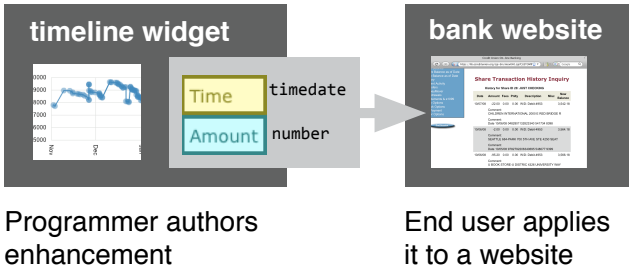

Figure 2: reform divides web enhancement into roles of authoring, for programmers, and attaching, for end users.

978-1-4244-4876-0/09/$25.00 ©2009 IEEE

novice end users, allowing them to scrape a variety of data layouts by example, without seeing the underlying webpage representation.

Describe progress to date

reform is a library for Firefox extensions. Rather than hard-code HTML or DOM patterns to access parts of a webpage, web enhancements (Firefox extensions) query the reform library with a schema expressing the general type of data they expect a webpage to contain. reform then prompts the user to click on parts of the page that match the schema, interactively training its scraper by example. For instance, a map enhancement will use reform to prompt the end user to click on example addresses. The reform library then generates and applies an extraction pattern, provides the enhancement with its requested integration points, and stores the pattern in a central database for future use by others.

3. Authoring database-backed sites by example

Explain why more work is needed

Traditionally, WYSIWYG tools have enabled non-programmers to develop webpages, but such tools were never powerful enough to support the modern data-driven database-backed websites that bring interactivity and mass two-way communication to the web. In future work we can use the web scraping technology in reform to allow non-programmers to create such database-backed websites by-example within WYSIWYG editors.

For example we can allow a non-programmer to implement the basics of a photo-sharing site like Flickr, a news site like CNN.com, or a social communication site like Facebook or Twitter. Non-programmers can currently implement a single page of such a site with a WYSIWYG editor like Dreamweaver, but to support dynamic user-defined data they must jump into the world of databases and server-side programming.

Describe plans for future research

I propose implementing a photo website like Flickr in the following way. First, the user makes a single concrete example HTML page that exemplifies the template he wants all pages to look like, and contains some example data laid out in the desired way. For the Flickr clone, for instance, he might copy and paste an existing Flickr page, or design his own. The page probably contains a header, navigation bar a list of photos with captions and tags, and a footer.

Now, the user needs to describe to the system which parts of this page are to come from database fields and which parts are static template pieces. This can be done with the reform web scraping technology. The user selects a photograph on the page and marks it as a "photo." The system learns a pattern and highlights everything else on the page that looks like a photo, and the user corrects this inference by providing positive and negative examples. Now the user selects a caption and marks it "caption," a username and marks it "username," and the set of tags and marks them "photo tags." After these patterns are described to the system, it is able to extract the example data on this page, deduce a database schema, and fill the database with the example data. It also records the HTML template code surrounding each datum, and how to map the template to the database it just induced. It can now reproduce a dynamic database-backed version of the website from the mockup, without requiring the user to drop into a programming language and understand how relational databases work.

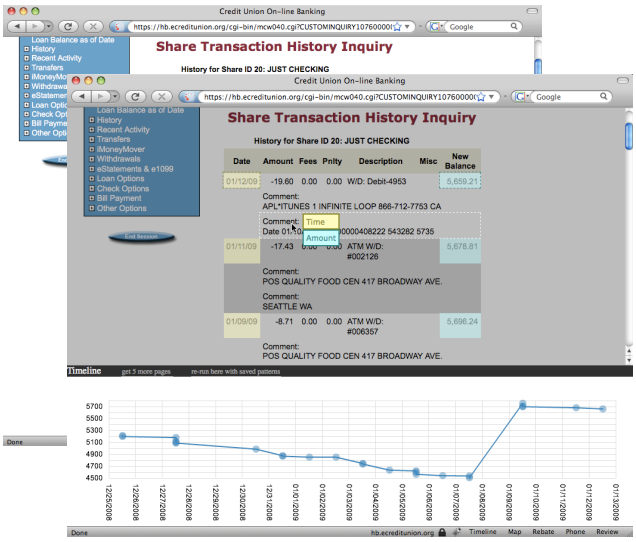

Figure 3: To plot her spending history with reform, the user clicks the *Timeline* button, then clicks an example *time* and *amount*.

4. Putting it together

Describe plans for future research

This *web authoring* technique can combine with the reform *web enhancement* technique to allow end users to modify existing websites to support new tasks. Suppose, for instance, that a Vietnamese rice trader wants to use Facebook to record and develop his trading business relationships. He does this by teaching reform what a person's name looks like on Facebook. Then he enters "edit" mode, and enters some comments about how reliable the other person was in business relationships. He selects his notes and marks it "trading notes," teaching reform a new database field. By copying this trading notes data to the other sections of the site where that user's name appears, the system learns how to display a contact's trading notes whenever the contact's name appears. He can then share this enhancement with other rice traders, allowing them to collaboratively track which traders have been reliable and are worth doing business with in the future.

There are far too many end users (1.4b), with far too many goals, on far too many websites (175m), to support with our existing labor force of programmers (~3m). This technology could lower the barrier between those who design the Internet and those who use it, and enable the 1.4 billion Internet end users to forge their own information systems.

5. References

1. Michael Toomim, Steven M. Drucker, Mira Dontcheva, Ali Rahimi, Blake Thomson and James A. Landay. Attaching UI Enhancements to Websites with End Users. *In Proc CHI 2009*. ACM Press (2009).