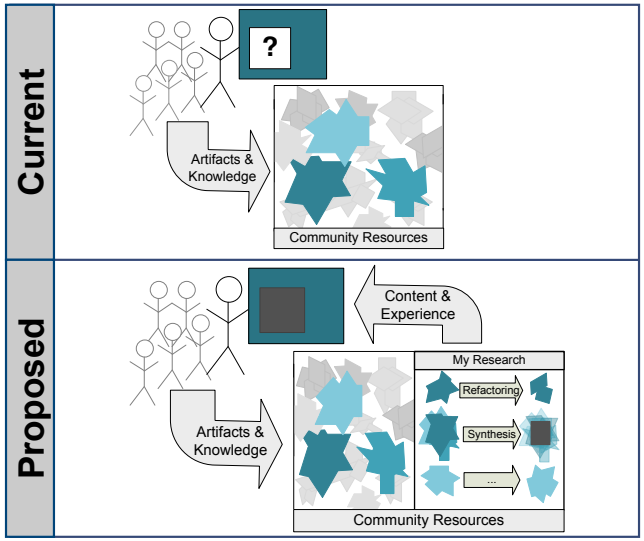
Towards Richer Resources for End User Programmers

Kathryn Stolee
ESQuaReD Software Engineering Group
University of Nebraska – Lincoln

Project Vision



Project Abstract

End users are already contributing large numbers of programs that impact society (tracking survivors of Hurricane Katrina, collecting polling data for presidential elections, writing apps for smart phones). With their domain knowledge and access to community resources, end users are creating programs to solve problems that are valuable and impactful to the community.

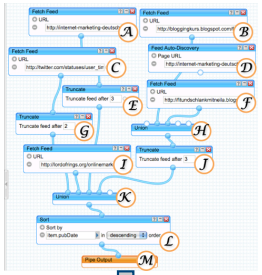
However, the community resources available to end users contain large amounts of diverse content, and the support provided to end users is insufficient for them to take full advantage of these resources. An end user has several needs when tapping the community for resources: 1) finding artifacts or information of interest, 2) selecting appropriate resources, 3) understanding selected resources, and 4) adapting the resources to fit their individual needs. We propose to use sophisticated analysis techniques to enable end users to tap into these rich community resources.

Current Work

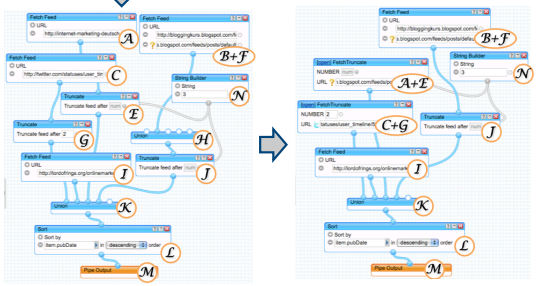
Mashups are becoming increasingly popular as end users are able to easily access, manipulate, and compose data from several web sources. We have observed, however that end user created mashups tend to suffer from several deficiencies.

In this work, we identify and specify smells that are indicative of those deficiencies, and design specialized refactorings to remove them. Our refactorings reduce the complexity of mashups, increase their abstraction, update broken sources of data and dated components, and standardize the mashup structures to fit community development patterns.

Motivating Example



For this study, we target smells found in pipe-like mashups, specifically targeting mashups created in the popular Yahoo! Pipes environment. Here, we illustrate what is a pipe-like mashup, what are the potential smells in such mashups, and how refactoring can help to remove those smells.



Methodology

Our methodology follows the steps of more traditional refactoring. We first identify smells – characteristics of pipes that are indicative of poor quality – and then devise refatorings – semantic preserving transformations – to address those smells.

Example Code Smells (and smell type)
- Unnecessary Module (*Laziness*)
- Duplicate Modules (*Redundancy*)
- Deprecated Module (*Environmental*)
- Global Isomorphic Path (*Population-based*)

Example Refactorings (target above smells)
- Remove Non-Contributing Module (*Reduction*)
- Merge Redundant Modules (*Consolidation*)
- Replace Deprecated Module (*Deprecations*)
- Extract Global Subpipe (P*opulation-based*)

We built a manipulation infrastructure to detect smells and apply transformations that target the smells. We scraped over 8,000 sample pipes from Yahoo!'s public repository, developed by thousands of end users, and performed an empirical study to identify the frequency of smells in the population and explore how many of those smells can be removed through refactoring.

Results & Future Work

Smells are present in 81% of the pipes we studied, and our proposed refactorings can reduce that number to 16%. This means the **refactorings could completely eliminate smells in 80% of the pipes** that previously had smells.

Future Work:
1. Explore whether end users understand that smells are 'bad'
2. Evaluate the utility of the refactorings in the hands of end users using a browser-based refactoring tool
3. Broaden the family of refactorings to address other smells we observed, but were not included in this study

Academic Advisor: Sebastian Elbaum
This work is supported in part by the EUSES Consortium through NSF-0915526, CFDA#84.200A: Graduate Assistance in Areas of National Need, and by the National Science Foundation through a Graduate Research Fellowship to Kathryn Stolee.