# Mashup Environments in Software Engineering

Lars Grammel, Christoph Treude, Margaret-Anne Storey
Department of Computer Science, University of Victoria
lars.grammel@gmail.com, ctreude@uvic.ca, mstorey@uvic.ca

## ABSTRACT
Too often, software engineering (SE) tool research is focused on creating small, stand-alone tools that address rarely understood developer needs. We believe that research should instead provide developers with flexible environments and interoperable tools, and then study how developers appropriate and tailor these tools in practice. Although there has been some prior work on this, we feel that flexible tool environments for SE have not yet been fully explored. In particular, we propose adopting the Web 2.0 idea of mashups and mashup environments to support SE practitioners in analytic activities involving multiple information sources.

## Categories and Subject Descriptors
D.2.6 [**Software Engineering**]: Programming Environments – *Integrated Environments*

## General Terms
Human Factors

## Keywords
Mashup, software engineering

## 1. INTRODUCTION
Too often, software engineering (SE) tool research is focused on creating small, stand-alone tools that address rarely understood developer needs. Even worse, many of those tools are evaluated in artificial laboratory settings, if evaluated at all. Thus, our understanding of the real developers' needs and of tool support to address them in industrial environments is fairly limited.

In contrast, SE practitioners implement their own pragmatic solutions using informal mechanisms such as dashboards [7]. When implementing these solutions, they are often unaware of related research, and vice versa, researchers are often unaware of solutions created by practitioners. Overcoming this unfortunate disconnect could help propel both SE research and SE practice. Studying these ad-hoc solutions could provide valuable insights into the needs of software developers and the design space of tool support, which could in turn lead to improved SE environments.


*Web2SE'10*, May 4, 2010, Cape Town, South Africa.


We propose that research should provide developers with flexible environments and interoperable tools, and we propose to study how developers appropriate and tailor these tools in practice. Although there has been some prior work on this, e.g. [7], we feel that flexible tool environments for SE have not yet been fully explored. Modern, plug-in based IDEs such as Eclipse and Jazz have been a major step towards this goal, and have already sparked many research projects that have found their way into daily SE work. However, lightweight automation of development tasks and flexible composition of different information sources would be useful in many daily SE activities.

This flexibility cannot be provided by plug-ins and development perspectives in current IDEs. In this paper, we envision how mashups could help with flexible and lightweight composition of SE information sources for analytical activities.

## 2. INFORMATION MASHUPS IN SE
Many activities in SE require searching, collecting and analyzing different kinds of information [2]. For example, in their daily code investigation tasks, software developers use a variety of information sources such as source code, issue trackers, and email [4]. The information sources are usually accessed through a variety of different tools and ways [4, 7]. This lack of integration puts the burden of integrating the information on the developer [4], and can lead to disorientation through thrashing [1] and to losing track of relevant information pieces [5]. Thus, there is a need for supporting the developers in collecting and analyzing information in an integrated, yet flexible fashion [4, 5, 6]. However, current tool support is limited even for combining different pieces of information from a single source [6].

We propose to adopt the Web 2.0 idea of mashups and mashup development environments [3] to support SE practitioners in analytic activities. Mashups are a lightweight approach of combining several data sources that are exposed as web-based services [3]. We believe that using appropriated mashup environments for analytical tasks in SE is beneficial, because such environments could enable developers to rapidly create task-oriented, situational information mashups just-in-time. Such light-weight mashups are easily modified as insights from the analyzed information are gained, which facilitates analytical activities. By having mashups as first class artifacts, they can be shared between developers, tailored by other developers to suit their individual preferences and reused in different circumstances.

We think that SE practice will benefit from using mashups by improved automation, collaboration, traceability, and documentation. For SE research, mashups are beneficial because they can provide a whole new set of artifacts that can be analyzed

to gain insights into a broad range of software engineering activities. This increased tangibility of SE process artifacts can help bridge the gap between research and industry.

## 3. RELATED WORK

Supporting developers in leveraging multiple information sources has recently received some attention by the SE research community. Holmes and Begel have developed Deep Intellisense, a tool for summarizing historical information about source code based on change sets, email, issue tracking data etc. [4]. However, while Holmes and Begel highlight the importance of flexibility of interaction, their tool cannot be tailored by the developers beyond arranging the layout of the three different views, and information seeking has to start from source code elements [4]. Fritz and Murphy created a tool that allows developers to answer questions by flexibly choosing different kinds of information and their composition order [2]. While this tool provides more flexibility, it is a single IDE widget and does not support storing its state or switching and combining different views such as timelines [2]. Based on their study of how developers seek, relate and collect information in software maintenance activities, Ko *et al.* propose creating a 'conceptual workspace' that supports collecting task-related information fragments and seeing them side-by-side [5]. We believe that such a flexible information workspace could be created by using mashup technologies, as we will outline next.

## 4. A MASHUP ENVIRONMENT FOR SE

We are developing an information mashup environment that enables developers to easily collect and analyze information from different data sources and thus aids their exploratory analytical activities. It will support the following features: IDE integration for adding information fragments; information visualization widgets; interactive construction of mashups using drag and drop; storing and sharing of mashups; export of configured widgets to dashboards; and tracking user interaction for insight provenance and traceability. A mockup scenario in which a developer explores which work items and which source code is related to the user interface polishing that took place before a major release is shown in. The arrows indicate one way of navigating the developer might use during the analysis.

The architecture of the information mashup environment is outlined in Figure 2. It is implemented using the Google Web Toolkit (GWT), and consumes data exposed by web-based APIs. By building adapters, other information sources can be used as well. The client part runs in modern web browsers. To provide IDE integration, we are planning to create a plug-in for the Eclipse platform.

## 5. STUDYING MASHUP ENVIRONMENTS

We believe that mashup environments, besides being useful for SE practitioners, would provide valuable data for SE researchers. For example, mining the mashup construction histories can bring insight into how programmers explore artifacts in analytical activities. We think that data mining and field studies of mashup environment usage provide promising avenues for future research. On a higher level, we need to research how to design SE mashup environments such that we can collect the data needed for these studies, and how security should be handled in SE mashup environments when integrating company-external resources.

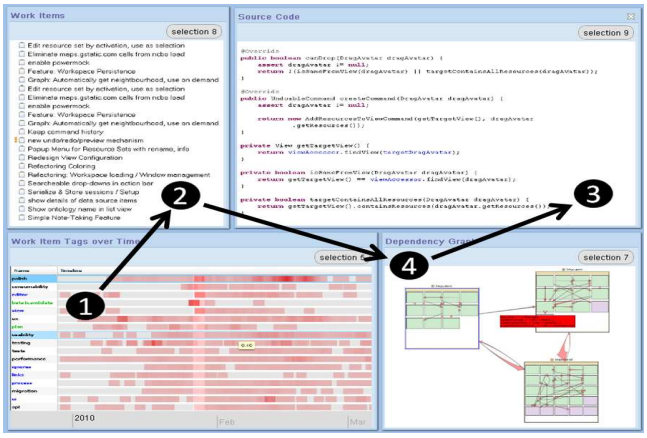


**Figure 1: Mockup scenario with work item tags over time (1), work items (2), source code (3) and dependencies (4)**

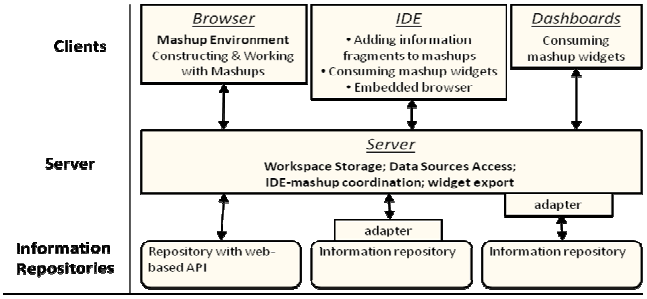


**Figure 2: Architecture of information mashup environment**

## 6. REFERENCES


[1] B. de Alwis and G. C. Murphy. Using visual momentum to explain disorientation in the eclipse ide. In *VLHCC '06: Proc. of the Symp. on Visual Languages and Human-Centric Computing*, pages 51–54, Washington, DC, 2006. IEEE.

[2] T. Fritz and G. C. Murphy. Using information fragments to answer the questions developers ask. In *ICSE '10: Proc. of the 32nd Intl. Conf. on Software Engineering*, New York, 2010. ACM. To appear.

[3] L. Grammel and M.-A. Storey. An End User Perspective on Mashup Makers. Technical Report DCS-324-IR, University of Victoria, September 2008.

[4] R. Holmes and A. Begel. Deep intellisense: a tool for rehydrating evaporated information. In *MSR '08: Proc. of the Intl. working Conf. on Mining software repositories*, pages 23–26, New York, 2008. ACM.

[5] A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Trans. Softw. Eng.*, 32(12):971–987, 2006.

[6] J. Sillito, G. C. Murphy, and K. De Volder. Asking and answering questions during a programming change task. *IEEE Trans. Softw. Eng.*, 34(4):434–451, 2008.

[7] C. Treude and M.-A. Storey. Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds. In *ICSE '10: Proc. of the 32nd Intl. Conf. on Software Engineering*, New York, 2010. ACM. To appear.