Educational Mashup in Example Authoring

I-Han Hsiao, Qi Li, Yi-Ling Lin, Hua Dong Wang
School of Information Sciences, University of Pittsburgh, USA
{ihh4, qil14, yil54, huw16}@pitt.edu

Abstract: Each educational resource management website relies on an authoring tool to provide example content. It takes time and experience for authors to create valuable content. Providing support during authoring can affect the quality and quantity of the examples. In this paper we focus on researching the issue of providing relevant examples in the process of example authoring.

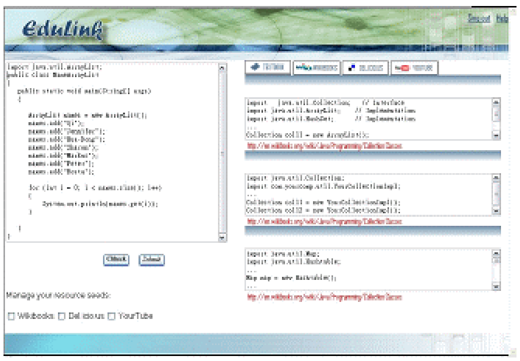
Keywords: Social Linking, Example Authoring, Mashup, Textbook Example, Wikibooks.

Introduction

Each educational resource management website relies on an authoring tool at the backend to provide content. The authoring tools are usually tailored to specific systems and require experienced authors or well-trained editors to provide valuable content. Most of the time, they require authors to use APIs or other references at hand to do the authoring. Inexperienced authors have to spend a lot of time to find more resources for authoring quality examples. In the context of program example authoring, in general, authors may search the internet for finding similar examples and reusing the partial codes. This can be a very time consuming process. It is also important to be able to validate examples found on the internet which requires validation functionality for the authoring tools. The availability of the similar examples from various sources and presented in different ways, the easiness of retrieval and the relevance of the examples affect the developing process and the quality and quantity of the examples. In our preliminary study of social linking to example authoring (Hsiao, Li & Lin, 2008), we proposed a mashup model to automatically link community wisdom to authors to ease various difficulties in authoring for developing programming language examples. Each mashup module dynamically pulls relevant examples for the users. The model aims to utilize the flexibility of mashups to increase the value of authoring tools. We found that diverse examples from Wikibooks, Del.icio.us and YouTube provided varied levels of information and support in terms of the content of examples and concrete senses in authoring. To further investigate the issue of example authoring, we hypothesize that providing relevant examples during authoring phase will facilitate the process of example authoring and help to author higher quality examples. To address this question, we focus on providing relevant examples in the process of example authoring.

1. System Description and Technologies of EduLINK

EduLINK (http://shtirlitz.exp.sis.pitt.edu:8080/EduLINK/) is an educational social linking system for example authoring. To offer authoring references and ease various difficulties in authoring, EduLINK automatically link community wisdom to authors. It evaluates users' authoring context and dynamically provides social examples from wikibooks, del.ici.ous, and youtube as well as the existing textbook example collection. Users could submit authoring context anytime during authoring to get relevant example support. EduLINK verifies content similarity and returns example codes and URLs to users. It has been deployed online, can be accessed anywhere anytime as users need.



```
import java.util.ArrayList;
public class NamesArrayList
{
   public static void main(String[] args)
   {

      ArrayList names = new ArrayList();
      names.add("[illegible]");
      names.add("[illegible]");
      names.add("[illegible]");
      names.add("[illegible]");
      names.add("[illegible]");
      names.add("[illegible]");
      names.add("[illegible]");

      for (int i = 0; i < names.size(); i++)
      {
         System.out.println(names.get(i));
      }

   }
}
```

```
import   java.util.Collection;   // Interface
import java.util.ArrayList;     // Implementation
import java.util.HashSet;       // Implementation
...
Collection coll1 = new ArrayList();
```

http://en.wikibooks.org/wiki/Java/Programming/Collection Classes

```
import java.util.Collection;
import com.yourcomp.util.YourCollectionImpl;
...
Collection coll1 = new YourCollectionImpl();
Collection coll2 = new YourCollectionImpl();
```

http://en.wikibooks.org/wiki/Java/Programming/Collection Classes

```
import java.util.Map;
import java.util.Hashtable;
...
Map map = new Hashtable();
...
```

http://en.wikibooks.org/wiki/Java/Programming/Collection Classes

Manage your resource seeds:

☐ Wikibooks ☐ Del.icio.us ☐ YouTube

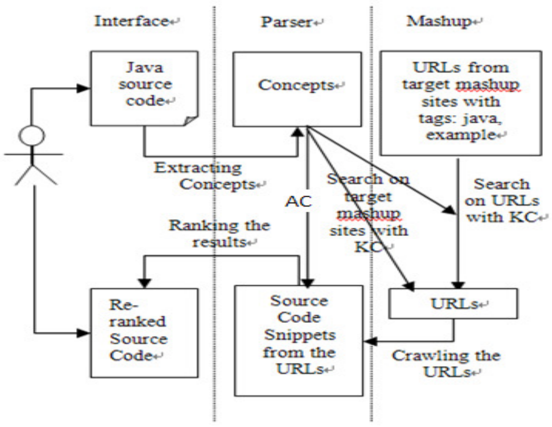


Figure 1. The main page of EduLINK system.   Figure 2. System Architecture

EduLink provides a web-based authoring interface. Figure 1 is the system screenshot. The left side of the screen is the authoring panel. Users can author the code in the authoring panel and check its programming syntax by clicking submit button anytime during authoring. Each mashup module is presented into a tab page on the right hand side. Each mashup module presents related example codes and source links on the basis of users' authoring content. The relative example codes are being displayed ordered by similarity ranking of users' authoring content. Users can access source links and get more detailed example descriptions in the pop-up window.

1.1 Process flow

The input of EduLINK is directly from users' example source codes in the authoring panel. The system includes a parser engine which performs two tasks. First, it examines the syntax of the example and catches the syntax error in the provided examples, if there's any. We define the caught concepts as KC (Key Concepts). The key concepts will be passed to each mashup module. Second, the parser extracts a list of concepts from the entire example, regardless the completeness of the example. We define the list of concepts as AC (Analyzed Concepts). AC will also be passed to each mashup module. We use the java ontology to parse the concepts, which was developed in (Henze et al., 2004). Once mashup module gets the KC from the parser engine, it starts to proceed mining on each mashup module. Each entry of results (source code snippet) generated from mashup module will be indexed by each on-site mashup parser. Then, the system will return ranked results to users by calculating the cosine similarity with the list of concepts generated and the AC derived from users. EduLINK will provide results for users according to every authoring validation or submission. Iterations of results are expected to assist users in example authoring.

1.2 Mashup Modules

- Textbook module covers three Java programming textbooks examples (Cay, 2008; Deitel& Deitel, 2007; Pohl & McDowell, 2006). Each example is documented into our database line by line with authors' original annotations if there're any. Upon each validation or submission from the user, textbook module will mine the whole example collection from the database. Return the ranked results by calculating the cosine similarity with the list of concepts generated from the example collection and the AC derived from users.
- Wikibooks (http://en.wikibooks.org/wiki/Main_Page) is a free content textbook and manual based on Wikimedia foundation wiki sets. Wikibooks has a complete Java book including plenty of Java source codes with detailed commands and usage explanations. It is a good source to provide the Java programming examples. Wikibooks Sample Codes Extracting Module (Wikibooks Module) is focused on extracting the examples codes according to the author editing Java source codes.
- Del.icio.us (http://del.icio.us/) is a collaborative tagging system for web bookmarks (Golder & Huberman, 2006). We discover the URLs contained "java" and "example" tags are mostly the links to java example code collections. Fewer users tag their URLs specifically anchored to an example. Therefore, in this mashup module, we grab the top 10 java example collection websites from del.icio.us and combine the key concepts to locate the example code snippets.
- YouTube is a video sharing website (http://www.youtube.com/). This module follows similar procedure as the way we mash del.icio.us site. We retrieve a stream of video clips tagged with terms such as "java" or "tutorial". Using Key Concepts and mining the video description and tags, we provide a narrowed-down list of videos.

2. Study Design

2.1 Participants

3 volunteers (1 female and 2 male) participated in the case study experiment and they were assigned to 3 groups randomly. The participants were all graduate students with former knowledge in Java programming language and advanced experience in programming. All the participants are familiar with computer operation and Internet browsing.

2.2 Dependent Variables

In this experiment, four dependent variables were involved and they were performance time, satisfaction, workload and usability.

- Performance time, the time needed for a participant to finish one single searching and authoring task.
- Satisfaction, was the scores obtained through a general satisfaction questionnaire on the scale of 1 to 7 (low to high satisfaction). It was a modified version of the satisfaction measure utilized by Cook & Choong (1996).
- Workload, it was measured by NASA-TLX Workload Scales (Hart & Staveland, 1988). The NASA-TLX Workload Scales mainly measures six workload aspects of users including mental workload, physical workload, time pressures, self-evaluate performance, degree of effort and degree of frustration.
- Usability, was the scores obtained through a general usability questionnaire on the scale of 1 to 5 (low to high usability). The questionnaire was a modified version of the System Usability Scale (SUS), which is a simple, ten-item scale giving a global view of subjective assessments of usability. (Brooke, 1986)

2.3 Independent Variables

This experiment focused on providing relevant examples in the process of example authoring. Therefore, the independent variable was: different resources of example content correlate to the relevant support during authoring, including the textbook examples, dynamic generated example passages from EduLINK and generic search from user queries.

2.4 Tasks and procedure

Subjects are from three different programming proficiency levels, beginner, middle, and professional. Beginner level has half year experiences with one semester classes; middle level has six years java programming experiences; and professional level is sun-certified java programmer. They are assigned to the same topic, ArrayList, using three different ways, Generic Search, Textbook Example and EduLINK. The task and authoring platform was issued by the random sequence. Each session of time spent was measured. Upon the completion of each task, the questionnaire was collected, including satisfaction, workload scales and usability forms.

3. Preliminary Results

Both the examples generated from EduLINK and textbooks helped users in authoring than generic search in terms of less authoring time (Table 3). However, results show that textbook examples helped user in 50% less amount of time than EduLINK did. On the other hand, in all the case studies, our ranking algorithm which served the relevant examples worked properly in helping users complete the tasks compared with generic task which confirmed our former hypothesis.

	Generic Search	**Textbook Example**	**EduLINK**
Beginner	21	6	18
Middle Level	9	7	13
Professional	14.5	7	11.75
Mean	14.83	6.66	14.25

Table 3 Performance Time According to Different Methods (Minutes)

Based on the past-experiment questionnaires, three levels subjects uniformly give high scores for EduLINK's usability. As the limitation of sample size, we cannot come to certain conclusion for the satisfaction and workload of the users regarding to the authoring task. However, the workload of professional user is surprisingly the highest whereas their satisfaction is the lowest, which we may explain as the professional user tends to have higher expectation for both themselves and their example quality.

According to the interview and observations, less experienced authors prefer having completed textbook examples as reference while authoring, as they claim that they have no sense of what characteristic should included for a qualified examples. There are no preferences for experienced authors. Additionally, experienced authors praised on EduLINK in providing relevant program passages, especially helpful for the higher level concept examples.

4. Summary and Future Work

In this study, the results show that completed and organized example codes from textbooks provided the best support for users in terms of less authoring time. With dynamically generated relevant examples from EduLINK, our subjects successfully completed the programming tasks and it outperformed than with generic search support merely. EduLINK reduces time to search for relevant example references online and helps authors to save time in searching and retrieve more precise results automatically. Our subjects also strongly agree that the example codes provided by our EduLINK mashup are helpful in supporting authoring, especially in authoring higher level concepts examples.

Although EduLINK could limit search sources more precisely than generic search online, the retrieved information based on dynamic concepts extracted from authoring context is not well-matching to users' need. In the future, we will focus on investigation about specifying the scope of the authoring example which may narrow down the search and match users' needs better. Our vision is to extend the scope to a broader range of purposes and users. For instance, involving with student authors and associate with students learning activities etc. We believe that our mashup model in authoring plan is promising and deserved to be carefully carried out and attention by educational communities.

Acknowledgements

We thank all the people who participated in this study. Special thanks to Rosta Farzan and Dr. Brusilovsky.

References

[1] Brooke J. SUS - A quick and dirty usability scale. Digital Equipment Co Ltd., Reading, United Kingdom. 1986

[2] Cay S. Horstmann (2008) Big Java, 3rd Edition, WileyPLUS.

[3] Cook J R. Cognitive and Social Factors in the Design of Computerized Jobs. Doctoral Dissertation. Purdue: Purdue University. 1991

[4] Deitel, H. M., & Deitel, P. J.. (2007).Java How to Program,7/e, Deitel & Associates Inc.

[5] Golder, S. & Huberman, B.A. Usage patterns of collaborative tagging systems. Journal of Information Science, Vol. 32, No. 2, 198-208 (2006)

[6] Hart S G and Staveland L E. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In: Hancock P.A. and Meshkati N. (Eds.) Human mental workload. Amsterdam: North-Holland, 1988. 139-183

[7] Henze, N., P. Dolog, and W. Nejdl, Reasoning and Ontologies for Personalized e-Learning in the Semantic Web. Educational Technology & Society, 2004. 7(4): p. 82-97.

[8] Hsiao, I., Li, Q. and Lin, Y. (2008) Educational Social Linking in Example Authoring, Hypertext 2008

[9] Pohl I. & MacDowell, C. (2006). Java by Dissection. Addison-Wesley.