Experience on Mashup Development with End User Programming Environment

Kwok-Bun Yue
Department of Computer Information Systems
University of Houston-Clear Lake
Houston, Texas, USA
yue@uhcl.edu

ABSTRACT

Mashups, Web applications integrating data and functionality from other Web sources to provide a new service, have quickly become ubiquitous. Because of their role as a focal point in three important trends (Web 2.0, situational software applications, and end user development), mashups are a crucial emerging technology for information systems education. This paper describes the result of a pilot experiment of an open-ended mashup assignment using an end user Web-based visual development environment: Yahoo's Pipes. Surveys, qualitative analysis, peer evaluations, and comparative analysis were used to assess the assignment. Initial results indicated that the assignment was effective, well received, and cost efficient. Students found it to be useful, interesting, appropriate, and of the right level of difficulty. They gained the needed expertise in mashups and Yahoo's Pipes within a short period of time. They developed mashup applications with the expected degree of complexity, maturity, and innovativeness. There were no logistical bottlenecks and grading the open-ended assignment appeared to be consistent among the instructor and peers. The peer evaluations were perceived by students as very useful, even more so than the actual mashup development. Although Yahoo's Pipes were in general well received, its limitations, such as the lack of programming capability, created some minor issues and changed the designs of some mashups slightly. IS educators interesting in integrating open-ended mashup assignments into their courses may consider including a robust peer evaluation component and selecting a mashup development environment that matches the assignment goals.

Keywords: Mashup, Web 2.0, situational software applications, end user programming, end user programming environment, Yahoo's Pipes, IS assignments, peer evaluations

1. INTRODUCTION

Mashups are Web applications that combine data or functionality from other Web sources into a new and integrated service (Wikipedia 2009a, Yu et al. 2008, Zang and Rosson 2008). They are expected to be developed quickly using open data sources or Application Programming Interfaces (API) (Zang and Rosson 2008). Their rapid initial successes, especially in using Google Map API, quickly fueled phenomenal development and adoption. Mashup "has become one of the hottest buzzwords in the Web application development area" (Yu et al. 2008). For example, the website programmableWeb (2009), which tracks mashups and related open APIs, recorded 4,254 mashups and 1,425 APIs. It also reported an increase of three mashups every day on the average.

The importance of mashups is not only in its ubiquity. It is also a focal point of three interlinked major trends in information systems: Web 2.0, situational software applications, and end user programming.

Since coined in 2004, Web 2.0 (O'Reilly 2005) has already become a household term. The pervasiveness of representative Web 2.0 applications, such as Facebook, Flickr, Twitter, Google Docs, and YouTube, ensures that the term has become a fundamental lexicon for the modern society. Besides being used universally, Web 2.0 also deeply influences nearly every facet of our lives: culture, education, business, technology, etc (Kim et al. 2009).

In particular, the importance of Web 2.0 cannot be underestimated in IS education. The Journal of Information Systems Education recently devoted a special issue with twelve papers on the effective uses of different Web 2.0 technologies, including blog, wiki, podcast, social network and virtual world, in IS education (Harris & Rea 2009). However, the impact of Web 2.0 is even deeper than the proper integration of Web 2.0 technologies into IS learning and teaching. Equally importantly, Web 2.0 profoundly affects core components of the subject knowledge of IS education: how software are conceived, planned, specified, designed, developed, updated, and used. Web 2.0 techniques, architectures, tools, standards, software development methodologies, design patterns, and project management approaches should be studied and assimilated into IS curriculum to complement the existing set of methodologies.

From this perspective, mashups are exemplary as an embodiment of Web 2.0 ideals (O'Reilly 2005, Kim et al. 2009). Mashups are highly popular and they frequently use Web 2.0 technologies such as AJAX, XML, RSS, JSON,

Open APIs, and Web data sources (Kim et al. 2009). Their application areas are diverse and closely associated with key Web 2.0 application domains such as social networks. Their development methodologies are representative of Web 2.0: rapid development and modification, crowd sourcing, extensive use of open standards and APIs, etc. Thus, mashups are excellent pedagogical vehicles for Web 2.0.

The second closely related trend is the proliferation of situational applications (SA), which loosely refer to applications built for addressing a particular situation, problem, or challenge (Cherbakov et al. 2007). Wikipedia (2009b) defines them to be "'good enough' software created for a narrow group of users with a unique set of needs." The application may be used specifically for a given task of a small social group (Shirky 2004), an enterprise business problem (Cherbakov et al. 2007), or any targeted situations. SAs are very useful because of their custom-made nature for particular situations. However, with limited user size, functionality, scope, and life-span, SAs can only be cost effective if their development cost is low enough. Until recently, this cost and benefit consideration did not favor SAs. Developing SA was just very expensive. The cost effectiveness balance had recently changed to largely favor SAs as their development cost was substantially lowered. Cherbakov (2007) listed eight contributing factors to the rapid rise of the popularity of Web-based SAs. These factors include general advances in the computing world: lower infrastructure costs, and advances in general computer literacy. They also include changes in the business world: increased requirements for business agility. However, the majority of these factors are related to the progress of Web 2.0: introduction of Web 2.0 technologies, Service-Oriented Architecture adoption, acceptance of community-based computing and collaboration, proliferation of APIs and Web components, and availability of numerous mashups and SAs in the public domain.

The rapid swelling of SAs represents special challenges to IS educations since SAs are quite different from traditional IS applications. SAs are supposed to be constructed in a quick-but-not-necessarily-dirty way in which simplicity, usability and accessibility are more important than function completeness and extensibility (Yu et al. 2008). Traditional component-based IS methodologies target professional software developers and do not necessarily work well with this new breed of applications which may be best developed by domain experts, not programmers (Cherbakov et al. 2007). From the perspective of SA, mashups are crucial. The purposes of the majority of mashup applications are mostly situational. In fact, Yu et al. (2008) stated that "mashup development differs from traditional component-based application development mainly in that mashups typically serve a specific situational (short-lived) need and are composed of the latest, easy-to-use Web technologies." Conversely speaking, a large class of SAs is Web-based and many Web-based SAs are mashups. Since mashups are interesting to many students, using them to introduce SAs into IS curriculum can be effective.

The third closely related trend is the ascension of end user programming and its development environment (Myer et al. 2006). In end user programming, software is driven, modeled, and developed by end users and not traditional programmers. End user programming is especially crucial for SAs since end users are domain experts of the situations and they know the business logic well to develop the software within the required short development life-cycle. Thus, Cherbakov et al. (2007) indicated that the new breed of SAs, "often developed by amateur programmers in an iterative and collaborative way, shortens the traditional edit-compile-test-run development life cycle."

This vast expansion of software developers to include a large class of end users resonate well with the underlying Web 2.0 concepts of community participation, crowd sourcing, and especially mass amateurization (O'Reilly 2005). Shirky (2004) compared the software development movement from programmers (experts) to end users (amateurs) with similar and inevitable movement in typing from secretaries (experts) to everyone (amateurs). However, software development is much more complicated than typing. Thus, the development of feature rich, easy to use, and domain-specific end user programming environments is crucial for the mass amateurization of programming.

Early mashup development, such as those involving Google Map API, was programming intensive. They were designed and developed by traditional programmers. More recently, much attention was devoted to end user programming environments for quick mashup development by non-programmers (Beletski 2008). Major companies provided Web-based mashup development platforms to encourage experimentation. Examples include Yahoo's Pipes (Yahoo 2009a), Google Mashup Editor (Google 2009), Microsoft's Popfly (Microsoft 2009), IBM's Damia (IBM 2009), and Intel's Mashmaker (Intel 2009). These tools provide easy to use visual environments for supporting to various degrees the different tasks of mashup development: identifying data sources, retrieving and parsing data from sources, and assembling data to create the desirable output of the mashups. Although not all of these efforts survived in standalone forms, their results were frequently absorbed into other main products.

It can be argued that the innovation of end user mashup development environment is at least as intensive in academia. Many end user programming environment prototypes have been built in academia based on different paradigms such as flowcharts, trees, and spreadsheets (Beletski 2008, Wang et al 2009). Examples include Marmite (Wong and Hong 2007), d.mix (Hartman et al. 2007), Mashroom (Wang et al. 2009), and Mashlight (Albinola et al. 2009). The vast interest in mashups and the subsequent innovation in mashup development tools are thus important drivers of the commercialization and research of end user programming. Mashups are in the forefront of end user programming. From this perspective, mashups are ideal for introducing end user programming and its environments into IS curriculum.

In summary, mashups are a crucial emerging technology at the center of three important IS trends and they are primed to be integrated into IS education. Although there are plenty of works on the uses of mashups in education, there is a lack of studies on mashup development in IS education. The purpose of this paper is thus to contribute in filling this gap by describing our experience gained through a pilot experiment on an open-ended graduate mashup assignment with peer evaluations using Yahoo's Pipes. Students were allowed to develop mashups of their own

choices in the open-ended project. We attempted to answer six questions that are probably interesting to like-minded IS educators:

1. Would the students be able to overcome the initial learning curve of the mashup development environment quick enough?
2. Would students be receptive to a mashup assignment in which no programming is needed?
3. Were open-ended mashup assignments interesting, useful and appropriate to the students?
4. Would the open-ended mashup assignment possess reasonably degrees of substance and complexity?
5. Could grading of such open-ended mashup projects be consistent?
6. Were peer evaluations valuable, and should they be included as an effective part of a mashup assignment?

The remainder of this paper is organized in the following manner. Section 2 provides background information about Yahoo's Pipes and why it was selected. Section 3 describes the experimentation setup, delineating the assignments, peer evaluations and surveys. Section 4 discusses the results of the experiment. We draw our conclusions in Section 5. Appendices are listed in Section 6.

2. YAHOO's PIPES

According to Yahoo (2009a), "Pipes is a free online service that lets you remix popular feed types and create data mashups using a visual editor." Pipes use a flowchart approach for building mashups (Wang et al. 2009) in which the Web-based visual editor provides developers a canvas to drag, drop and use preconfigured Pipes modules to compose the mashup. Yahoo (2009b) currently provides 53 modules in nine categories: Data Sources, User Input, Operator, URL, String, Date, Location, Number and Deprecated. Each module serves a single function and output from one module can be wired as an input to another module. The pipes of chained modules are eventually fed into an output module for deployment in the Yahoo's Pipes site, which hosts tens of thousands of Pipes. As an example, Figure 1 shows a basic Pipes mashup for depicting current weather information of a user specified weather station ID, such as KRIC at Richmond International Airport, in a map.

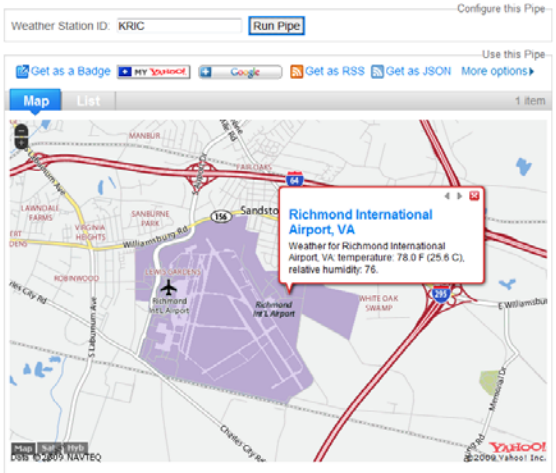

Figure 1. A Basic Weather Information Pipes Mashup

The internal Pipes layout for the basic weather mashup looks like a flowchart and is shown in Figure 2. Weather station id is obtained from the user using a User Input module (module #1) to construct a string with a String Builder module (module #2), which is then used to compose an URL (module #3). The URL is used to fetch the weather information in a custom-designed XML format from National Oceanic and Atmospheric Administration's National Weather Service (2009) (module #4) The geo-location of the retrieved XML weather information is determined using a Location Extractor module (module #5). Since the mashup output format is Really Simple Syndication (RSS 2009), selected key elements are copied and renamed from the input XML content using a Rename module (module #6). In particular, the <description> output RSS element is prepared by using regular expressions in the Regex module (module #7) so that it can be used as the content of the map's popup window. The RSS output is then fed to the Pipe Output module (module #8). On executing the mashup, Yahoo's Pipes determines the existence of geo-location information and automatic displays a Yahoo's map for the mashup.

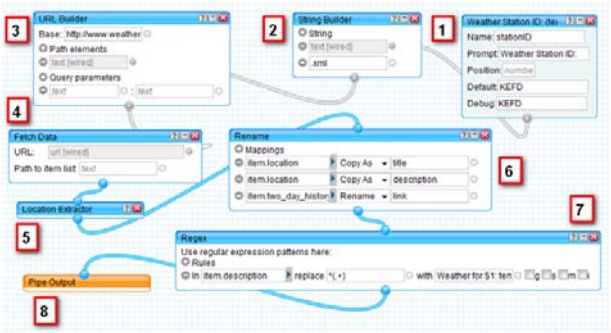

Figure 2. Layout of the Basic Weather Pipes Mashup

It can be seen that no programming is involved in the construction of the mashup. The use of regular expressions amounts to one of the more stringent expertise required for Pipes development. Thus, Yahoo's Pipes fits the description of end user development well. After studying similar commercial mashup tools briefly, we eventually selected Yahoo's Pipes for our assignment. Unlike other enterprise mashup application products, Yahoo's Pipes is free. It is reasonably popular, with tens of thousands of Pipes hosted. The learning curve appears to be acceptable. Its visual editor is a purer end user programming environment as programming is not even supported. We were interested in investigating how students response to such a purer system.

3. EXPERIMENT SETUP

The graduate course XML Applications Development in our university is usually taken by majors of Computer Information Systems, Software Engineering and Computer Science. The course provides an overview of XML standards and technologies, including XML, XML servers, XML editors, XML databases, Document Type Definition (DTD), XML Schema, XML parsing using Document Object Model (DOM) and Simple API with XML (SAX), XML Style Sheet (XSL), XSL-Transformation (XSLT), and XQuery.

We conducted our experiment in a nine week course during the summer semester of 2009. The students were required to complete five homework assignments. One assignment was a brief term paper in an approved XML topic. The other homework assignments involved software development using XML parsers, remote data sources, and XSLT. The Yahoo's Pipes assignment was the third homework assignment and thus represented about one fifth of the student work.

The first assignment asked the students to develop a simple XML server to serve news information on a user specified topic in a predefined XML format. The original news headings were obtained and extracted from two different data sources in RSS 2.0 (which is in XML format) and Comma Separated Value (CSV) respectively. Thus, although the concept of mashup had not formally been introduced in the class before the Pipes assignment, students already gained experience in integrating remote data sources for situational applications.

The Yahoo's Pipes Mashup assignment had two phases. In the first phase, students were required to design and implement a Yahoo's Pipes application of their choices, with the emphasis that the application should be innovative, useful, and significant. There was no restriction on the nature and application areas of the Pipes applications. We decided early on that open-ended projects were more appropriate for mashup applications. After all, mashups are ideal for open innovation by mixing publicly available data sources. Furthermore, if every student worked on the same Pipes application, plagiarism can be an issue since Pipes is searchable and its internal layout is public. However, grading would be more difficult for open-ended assignments and we included mechanism to investigate whether grading could be consistent. The result will be discussed later in the paper.

The students also needed to turn in a report on their Pipes applications with five sections:

1. Data sources used
2. Data extraction and processing procedure
3. Yahoo's Pipes modules used
4. Screenshots
5. Discussion

The discussion section included three sub-sections: difficulties encountered, Pipes experience gained, and mashup experience gained.

After the completion of the first phase, each student was asked in the second phase to evaluate five randomly selected peer applications using a standard template (Appendix 1). They rated their peer work on a Likert scale from 1 (worst) to 5 (best) and provided the rationale for the ratings on five aspects of the applications: innovativeness, usefulness, interestingness, design and implementation, and overall applications. Finally, they answered the question; "If you were the instructor of this course, what grade will you assign (from 0 to 100)? Why?" A broad perspective is crucial for successful mashup development. We felt that peer evaluations would encourage students to study other mashup applications systematically to gain a panorama view.

To provide the necessary background, a one hour lecture on mashup and Yahoo's Pipes development was provided. This included a brief discussion on the definition, nature and landscape of mashup development, followed by a live demonstration of developing from scratch the basic weather Pipes mashup described in Section 2 using the Web-based Pipes visual editor.

To gauge students' initial perception and experience on mashups in general, and Yahoo's Pipes in particular, a General Mashup Survey (Appendix 2) and a simple Yahoo's Pipes Survey (Appendix 3) were conducted before the lecture. To measure changes in perception, students repeated the General Mashup Survey again after they turned in their peer evaluations. At this point, students also completed a more detailed survey on their Pipes experience on the assignment (Appendix 4 Pipes Assignment Survey.)

4. RESULTS

It is important to note that there were only eight students in the class so any result of this pilot experiment should be considered preliminary. However, interesting perception and patterns can already be observed and the initial result is promising.

4.1 Yahoo's Pipes Development

The Pipes assignment was open in nature with no strict guideline on complexity, substance, and areas of applications. To analyze the appropriateness of the assignments, several characteristics were measured. Table 1 shows the results obtained by studying the Pipes flowcharts and submitted Pipes reports.

Characteristics	Student Pipes		
	Min	Avg	Max
C1. # of Yahoo's data sources	0	0.75	1
C2. # of other data sources	0	1.63	3
C3. Total # of data sources (C1 + C2)	1	2.38	4
C4. # of data feed	1	3.5	7
C5. # of kinds of Pipe's modules used	5	10	13
C6. # of Pipe's modules	5	18.8	29
C7. # of user input fields	0	2.13	6

Table 1. Characteristics of Student Pipes Mashups

The Yahoo's Pipes editor provides five built-in modules for fetching from selected data sources: Yahoo! Search, Yahoo! Local, YQL, Google Base and Flickr. They are the easiest to use. Characteristic C1 of Table 1 measured the number of Yahoo's data sources used. To use other data sources (Characteristic C2), developers need to investigate the data format, construct the appropriate URL to fetch the source, and parse the content. They are thus more complicated. Characteristic C3 sums up C1 and C2. Since a mashup can use more than one data feeds (Characteristic C4) from the same data source, C4 can be larger than C3. Characteristic C5 measures how many of the available 53 Pipes modules were used. Since each kind of modules could be used more than once, the number of Pipes modules used (Characteristics C6) could be more than C5. Finally, Characteristic C7 measures the number of user input fields.

We performed a simple comparative analysis on a random collection of 20 Yahoo's Pipes selected from the top 100 results of searching for the string "*" in the Yahoo's Pipes website. This provides a comparison framework to study the relative substance and complexity of the student

works. Pipes flowcharts, such as the one shown in Figure 2, are public. The selected Pipes were studied to compile Table 2 as a comparison with the student Pipes.

Characteristics	Average of Student Pipes	Average of 20 Random Pipes
C1. # Yahoo's data sources	0.75	0.1
C2. # Other data sources	1.63	1.7
C3. Total # of data sources	2.38	1.8
C4. # of data feed	3.5	1.8
C5. # of kinds of Pipe's modules used	10	6.7
C6. # of Pipe's modules	18.8	9.25
C7. # of user input fields	2.13	2.15

Table 2. Average Characteristic Scorings of Students Pipes Compared to Twenty Randomly Selected Yahoo's Pipes

Besides these characteristics, we also tracked whether a map was used in the mashup output (Characteristic C8) and whether regular expressions (Characteristic C9) were used to construct the Pipes flowchart. The results are shown in Table 3. We reasoned that the use of regular expressions is a good indicator of complexity.

Characteristics	Percentage used in Student Pipes	Percentage used in Random Pipes
C8. Use of maps in output	75%	5%
C9. Use of regular expressions	50%	25%

Table 3 Use of maps and regular expressions

It can be seen that student Pipes are significantly more complicated and substantial than randomly selected Pipes in nearly all characteristics. The differences are especially prominent in characteristics C1 (# of Yahoo's data sources), C4 (# of data feed), C6 (# of Pipes modules) and C9 (uses of regular expressions). More detailed analysis would put the average student Pipes in the top quartile of complexity with respect to the randomly selected Pipes. The elevated complexity is appropriate as the students were developers with information systems and computing background whereas the expected general authors of Yahoo's Pipes were a mix of end users and developers. The complexity of Yahoo's Pipes mashups is limited by their natures as situational applications, and by the restrictions of the Pipes platform. Thus, we reasoned that the complexity and substance of the homework assignment was appropriate to represent one fifth of the student work, even though there were no programming activities.

It is worthy to point out some interesting patterns. The randomly selected Yahoo's Pipes only used 0.1 Yahoo's data sources (Characteristic C1) on the average, as compared to 0.75 of the student Pipes. It is likely that the tens of thousands of Pipes in the Yahoo's site consumed the mixing possibilities of the five built-in Yahoo's data sources quickly. As a result, the majority of Pipes mashups used other external data sources. On the other hand, students were not asked to study other Pipes before developing their own. They were not affected by existing Pipes that used built-in Yahoo's data sources. They might thus tend to use Yahoo's data sources more frequently because of their ease of use. Adding an additional phase for evaluating existing mashups before actual development may be beneficial for developing more unique Pipes, and it will be a target of future study.

Perhaps the most significant difference was the use of maps (75% in student Pipes and 5% in randomly selected Pipes). The low percentage of map applications in the randomly selected sample is somewhat surprising since it is generally agreed that mapping applications account for a large percentage of mashups. In fact, early successes of mapping applications such as Chicago Crime Map (2009) and housingmaps.com (2009) contributed to a large degree to the initial popularity of mashups. About 35% of all mashups cataloged in the website programmableWeb (2009) were tagged as mapping applications. Since our sole classroom Pipes demonstration was a mapping mashup, it might skew more mapping mashup development and thus student Pipes were more in line with the percentage of mapping applications recorded in programmableWeb.

One reason for the relatively low percentage of mapping applications in Yahoo's Pipes might be that many authors used them as embedded mashups to extract data lists and images for their own websites, blogs or social networks through a mechanism called Yahoo's Badges (Yahoo 2009c). These embedded mashups are even more situational in nature and tend not to use maps a lot. As mashups become more ubiquitous, mashups as an embedded component will be even more common. Embedded mashups as assignments are thus a target of our future study.

4.2 Pipes Reports

Students turned in reports to accompany their mashup applications. The lengths of their reports ranged from a minimum of 409 words to 1,626 words, with the average being 1,007 words. Overall, they matter-of-factly cataloged their mashups in the first four sections: data sources, data extraction and processing methods, Pipes modules used, and screenshots. On the other hand, the technical discussion in the last section displayed much more interesting variety and insight.

The number of technical difficulties delineated in the student reports ranged from 0 to 3, with an average of 1.5. Table 4 summarizes the categories of technical difficulties elaborated.

Technical Problems	Counts
T1. Finding appropriate data sources	3
T2. Limiting Pipes framework (including module library)	2
T3. Inadequate Pipes documentation	2
T4. Getting and parsing appropriate data from data sources	2
T5. Pipes editor runtime problem	1
T6. Usage of Pipes modules	1
T7. Technical problems in the application domain	1

Table 4. Technical Problems Described in Student Reports

The main technical difficulties were associated with data sources (T1 and T4) and the Yahoo's Pipes platform (T2, T3, T5 and T6). Only one was related to application logic (T7). None of these technical problems were overwhelming and they were eventually overcome, some with ease. In two instances, the original designs changed slightly because of problems in data sources and the lack of programming capability in Pipes. No students reported that the learning curve was too steep. This result suggested that the level of difficulties of the assignment might be appropriate.

Students also elaborated on their Pipes experiences, which are summarized in Table 5.

Cited Yahoo's Pipes Merit or Drawback	Counts
P1. Limiting framework (including module library)	4
P2. Easy development	4
P3. Decent visual development	3
P4. Strong capability	3
P5. Lack of programming capability	3
P6. Inadequate documentation	1
P7. Inadequate data sources	1

Table 5. Pipes Experience Described in Student Reports

Yahoo's Pipes generally gained rather warm reception as an easy-to-use visual mashup platform with strong built-in capabilities (P2, P3 and P4 of Table 5). This spoke well for its use as the tool for a relatively small assignment. One student summarized this well: "The interface is very nice and easy to use, and the visualization of the flow of data through various functions makes the application not only easy to understand but logical and enlightening. I really enjoyed this assignment very much." Another student suggested assigning Pipes homework to introductory undergraduate computing courses since the visual flowchart can help beginners to grasp how data is acquired and processed to generate the required output.

However, not everyone was all rosy about Pipes. Limitations of the Pipes platform (P1) and lack of programming capability (P5) had been cited to have constrained the scope and depth of their Pipes applications. One student was especially adamant in the view that the lack of programming capability has rendered Pipes "almost useless". Selecting a more flexible mashup with programming capability may mitigate these constraints but can significantly increase the assignment complexity. For example, Joomla (2009), an open source Content Management System (CMS), can be regarded as a kind of mashup environment where Joomla components using different data sources can be mixed to create a web page. Yue, et al. (2009) described using Joomla to develop domain-specific social network websites with and without programming as capstone projects. However, these were semester long projects. The effective use of programmable end user mashup platforms in relatively short assignments is an issue for future study.

4.3 Peer Evaluations

The purpose of peer evaluations was to spur students to carefully study other works to acquire a broader perspective on mashups. Creativity does not work in vacuum and exposure to diverse ideas on the same subject area facilitates innovation. This peer investigation can help students gaining insight on mashup patterns (Wong & Hong 2008), which are crucial in effective application formulation. Each student evaluated five randomly selected peer works using a standard template (Appendix 1). Not counting the words in the template, the lengths of peer evaluations ranged from a minimum of 83 words to 470 words, with the average being 199.6 words. Thus, each student wrote a total of about 1,000 words on the average, a reasonable length with respect to the nature of the evaluation assignment. Furthermore, qualitative analysis of the peer evaluations indicated that students provided reasonable level of insight in their comments, many of which are parallel to those made by the instructor.

Table 6 summarizes the average assessment score of the peer evaluations in a Likert scale from 1 (worst) to 5 (best). The result was not unexpected except for the interesting observation that the overall grades were more generous than individual quality assessments.

Mashup's Quality	Score (1 to 5)
Innovativeness	3.83
Usefulness	3.65
Interestingness	3.60
Design and implementation	3.66
Overall quality	3.85
Grade (0 to 100)	91.75

Table 6. Average Peer Evaluation Result

A question we set out to investigate was whether there would be consistency in assessing and grading an open-ended end user development project like ours. To do so, we ranked the eight student works using peer evaluations from 1 to 8. We also ranked the student works using grades assigned by the instructor. A preliminary analysis of the correlation of these rankings indicated that the instructor and peer evaluations were in general in agreement.

Overall, these preliminary results show that consistent assessment of open-ended mashup assignments is attainable among different evaluators, especially if the scope and methodology are clearly specified.

4.4 Mashup Survey Results

The same General Mashup Survey (Appendix 2) was conducted before and after the assignment. A Yahoo's Pipes background survey (Appendix 3) was also conducted before the assignment. In term of background in mashups, no student had developed any general mashups application or Pipes application beforehand. Before the assignment, the average expertise level of mashups experience in a Likert scale of 5 (1: know nothing to 5: expert) was reported as 1.50, with the majority knew nothing about mashups. The highest claimed mashup expertise level was 3. This lack of experience was even more prominent in Yahoo's Pipes as only one student claimed some cursory prior knowledge. After the assignment, the claimed average expertise level rose to 3.56, with one student indicated an expert knowledge level of 5. This initial lack of prior knowledge and the subsequent significant increases in self-perceived expertise level suggested that the assignment was both effective and appropriate. The learning curve appeared to be reasonable.

The changes in average perception on mashup before and after the assignment were summarized in Table 7.

Mashup Applications Perception	Before	After
Usefulness (1 not useful at all, 3 neutral, 5 very useful)	3.86	4.13
Interestingness (1 not interesting at all, 3 neutral, to 5 very interesting).	4.14	4.13
Difficulty in development (1 very difficult, 3 neutral, to 5 very easy).	2.75	3.44
Appropriateness as assignments (1 not appropriate at all; 3 neutral, to 5 very appropriate)	4.43	4.38

Table 7. Changing Mashup Perception in Survey before and after the Assignment

In the two surveys, students maintained the perception that mashup applications were interesting and that it was very appropriate to use them as assignments. Importantly, after the assignment, their perceptions on the usefulness and ease of development significantly improved. Both observations signified the attractiveness of the assignment and the potential of rapid gain in mashup expertise.

4.5 Course Survey Results

After the assignment, students also completed a course survey (Appendix 4) to reflect on course related aspects of the assignment. The median student reported to have spent about five hours to learn Pipes after the lecture as preparation and ten hours to actually develop the mashup application. Three students indicated that they were immediately ready after the lecture. The longest time reported was 40 hours for the whole assignment. This seems to be a reasonable amount of time for an assignment of this nature. Table 8 summarizes the average perceived assignment characteristics.

Assignment	Interest-ingness	Usefulness	Appropriate-ness
Part 1 Development and Report	4.38	3.75	4.00
Part 2 Peer Evaluations	4.00	4.13	4.13
Overall	4.25	3.94	4.00

Table 8 Perceived Assignment Characteristics

Overall, the survey results indicated that the assignment was well received. We expected the students to be more interested in mashup development than peer evaluations, and the survey confirmed this anticipation. Peer evaluations were added as a part of the assignment based on their usefulness in providing a wide view of the subject matter. We hoped that the student might agree to their usefulness, even if they might not be as interesting. It turned out that the students more than agreed and rated peer evaluations as even more useful than mashup development. After spending hours to gain insight on their own development, it is likely that the students found investigating mashups of different aspiration, formulation, designs, approaches, and techniques strongly enriched their freshly gained experience. This result thus suggested that adding a robust peer evaluation component to open-ended assignments may be rather beneficial.

The remaining results of the course survey are similar to the results reported earlier and they are skipped here.

5. CONCLUSIONS AND FUTURE WORK

This paper presents our pilot experiment on an open-ended mashup assignment using Yahoo's Pipes. Mashup is a crucial emerging technology for IS education and the initial result was promising. It indicated that such assignment can be cost effective. The preliminary results show positive answers to all six questions we set out to investigate. Students show no major difficulty in using a visual end user programming environment and they gained expertise in mashup development quickly. A robust peer evaluation component was perceived to be highly useful for the open-ended project.

In the future, we would like to expand the experiment in several directions. This includes embedded mashups, and the use of an alternative mashup development environment that allows programming to fit the course goals. The assignment setting may also be modified to include a study of existing mashups before the actual application development. We would consider the addition of some restrictions on the open-ended project. We would also like to study how mashups can effectively be integrated into various IS courses in different curricula, such as the model curricula for the undergraduate program, IS 2002 (Gorgone et al. 2002), and the graduate program, MSIS 2006 (Gorgone et al. 2006).

6. ACKNOWLEDGEMENTS

I would like to thank our students, the anonymous reviewers, and the assistant editor for their invaluable input.

7. REFERENCES

Albinola, M., Baresi, L., Carcano M., and Guinea, S. (2009), "Mashlight: a Lightweight Mashup Framework for Everyone." 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009) at 18th International World Wide Web Conference (WWW 2009) Madrid, Spain, April 20-24, 2009.

Beletski O. (2008), "End User Mashup Programming Environments." Retrieved August 10, 2009 from http://www.tml.tkk.fi/Opinnot/T-111.5550/2008/End%20User%20Mashup%20Programming%20Environments_p.pdf.

Cherbakov, L., Bravery, A. and Pandya, A. (2007), "SOA meets situational applications, Part 1: Changing computing in the enterprise." IBM developerWork, retrieved August 17, 2009 from http://www.ibm.com/developerworks/webservices/library/ws-soa-situational1/?S_TACT=105AGX01&S_CMP=LP

Chicago Crime Map (2009), "EveryBlock's Chicago crime section." Retrieved August 10, 2009 from http://chicago.everyblock.com/crime/.

Google (2009), "Mashup Editor Blog." Retrieved August 20, 2009 from http://googlemashupeditor.blogspot.com/2009/01/from-mashup-editor-to-app-engine.html

Gorgone J., Davis, B., Valacich, S., Topi, K., Feinstein, D. and Longenecker, H. (2002), "IS 2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems." Retrieved October 7, 2008 from http://www.aisnet.org/Curriculum/IS2002-12-31.doc.

Gorgone J., Gray P., Stohr E., Valacich J. and Wigand R. (2006), "MSIS 2006: model curriculum and guidelines for graduate degree programs in information systems." SIGCSE Bulletin, Vol. 38, No. 2, pp. 121-196.

Harris, A. and Rea, A. (2009), Web 2.0 and Virtual World Technologies: A Growing Impact on IS Education, Journal of Information Systems Education, 20 (2), pp 137-144.

Hartmann, B., Wu, L., Collins, K. and Klemmer, S. (2007), "Programming by a sample: rapidly creating web applications with d.mix." Proceedings of the 20th Annual ACM Symposium on User interface Software and Technology, Newport, Rhode Island, October 07 - 10, 2007, pp 241-250

Housingmaps.com, "Housingmaps.com Home Page." Retrieved August 10, 2009 from http://www.housingmaps .com/.

IBM (2009), "Damia", Retrieved August 20, 2009 from http://services.alphaworks.ibm.com/graduated/damia.html.

Intel (2009), "Mashmaker." Retrieved August 20, 2009 from http://mashmaker.intel.com/web/

Joomla (2009), "Joomla's home page." Retrieved August 15, 2009 from http://www.joomla.org/.

Kim, D., Yue, K., Hall, S. & Gates, T. (2009), Web 2.0 Technologies, Principles, and Applications: Global Diffusion of the Internet XV: Web 2.0 Technologies, Principles, and Applications: A Conceptual Framework from Technology Push and Demand Pull Perspective46, Communications of the Associations of Information Systems, Vol. 24, pp 657-672.

Microsoft (2009), "Popfly." Retrieved August 20, 2009 from http://www.popfly.com/.

Myers, B.A., Ko, A.J., and Burnett, M.M. (2006), "Invited research overview: end-user programming." Proceedings of CHI Extended Abstracts, Montreal, Quebec, Canada, April 22-27, 2006, pp. 75-80.

National Oceanic and Atmospheric Administration's National Weather Service (2009), "XML Feeds of Current Weather Conditions." Retrieved August 9, 2009 from http://www.weather.gov/xml/current_obs/.

O'Reilly (2005), "What Is Web 2.0? Design Patterns and Business Models for the Next Generation of Software." Retrieved August 18, 2009 from http://www.oreillynet .com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

ProgrammableWeb (2009), "Mashups Tag Searching." Retrieved August 10, 2009 from http://www. programmableweb.com/tag/mapping.

RSS (2009), "RSS 2.0 at Harvard Law." Retrieved August 9, 2009 from http://cyber.law.harvard.edu/rss/rss.html.

Shirky C. (2004),"Situated Software." Retrieved on August 17, 2009 from http://www.shirky.com/writings/situated _software.html.

Wang, G., Yang, S., and Han, Y. (2009). "Mashroom: end-user mashup programming using nested tables." Proceedings of the 18th international Conference on World Wide Web (WWW '09), Madrid, Spain, April 20-24, pp. 861-870.

Wikipedia (2009a), "Mashup (web application hybrid)" Retrieved August 17, 2009 from http://en.wikipedia.org/ wiki/Mashup_%28web_application_hybrid%29.

Wikipedia (2009b), "Situational application" Retrieved August 20, 2009 from http://en.wikipedia.org/wiki/ Situational_application

Wong, J. and Hong, J. (2007), "Making Mashups with Marmite: Re-purposing Web Content through End-User Programming." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, California, April 28 - May 03, 2007, pp 1435-1444.

Wong, J. and Hong, J. (2008), "What do we "mashup" when we make mashups?" Proceedings of the 4th international workshop on End-user software engineering: WEUSE 08, Leipzig, Germany, May 12, pp. 35-39.

Yahoo (2009a), "Yahoo's Pipes documentation page." Retrieved August 9, 2009 from http://pipes.yahoo.com /pipes/docs.

Yahoo (2009b), "Yahoo's Pipes Editor." Retrieved August 9, 2009 from http://pipes.yahoo.com/pipes/pipe.edit.

Yahoo (2009c), "Yahoo's Pipes Badges." Retrieved August 10, 2009 from http://pipes.yahoo.com/pipes/badgedocs.

Yu, J., Benatallah B., Casati F. and Daniel, F. (2008), "Understanding mashup development." IEEE Internet Computing, 12(5), pp. 44-52.

Yue, K., De Silva, D, Kim, D., Aktepe, M., Nagle, S., Boeger, C. Jain, A, & Verma, S. (2009), "Building Real World Domain-Specific Social Network Websites as a Capstone Project." Journal of Information Systems Education, 20(1), pp. 67-76.

Zang, N. and Rosson, M. (2008), "What's in a mashup? And why? Studying the perceptions of web-active end users," IEEE Symposium on Visual Languages and Human-Centric Computing. VL/HCC, Herrsching am Ammersee, Germany, September 15-19, pp.31-38.

AUTHOR BIOGRAPHY

Kwok-Bun Yue (B.S., M.Phil., Chinese University of Hong Kong, M.S., Ph.D., University of North Texas) is a Professor of Computer Information Systems and Computer Science at University of Houston-Clear Lake (UHCL). He is the chairperson of the Division of Computing and Mathematics. His research interests are in Internet computing, semi-structured data, and information systems and computer science education. He had published more than 30 technical papers. Dr. Yue was a recipient of the UHCL Distinguished Teaching Award, the UHCL Piper Award and the UHCL Fellowship Award, and had served as a CTO of a startup company.

APPENDICES

Appendix 1 Peer Evaluation Template

Rank the application from 1 (worst) to 5 (best) on the following categories. Provide justification.

1. How do you rate the innovativeness of the application (1 to 5)? Why?
2. How do you rate the usefulness of the application (1 to 5)? Why?
3. How do you rate the interestingness of the application (1 to 5)? Why?
4. How do you rate the design and implementation of the application (1 to 5)? Why?
5. How do you rate the overall application (1 to 5)? Why?
6. If you were the instructor of this course, what grade will you assign (from 0 to 100)? Why?

Appendix 2 General Mashup Survey

Scale: 1 (negative answer) to 5 (positive answer)

1. How familiar are you with Web Mashup applications? (scale from 1 to 5: 1 know nothing; 3 average; 5: expert).
2. Have you developed Mashup Applications? Yes or no? If yes, how many Mashup applications have you developed?
3. How useful do you think Mashup Applications are? (scale from 1 to 5: 1 not useful at all; 3 neutral; 5 very useful).
4. How interesting do you think Mashup Applications are? (scale from 1 to 5: 1 not interesting at all; 3 neutral; 5 very interesting).
5. How difficult do you think Mashup Applications are? (scale from 1 to 5: 1 very difficult; 3 neutral; 5 very easy).
6. Do you think Mashup Application development assignments are appropriate for CIS/CS/SWEN students? (scale from 1 to 5: 1 not appropriate at all; 3 neutral; 5 very appropriate).

Appendix 3 Yahoo's Pipes Survey

1. How familiar are you with Yahoo's Pipe Mashup applications? (scale from 1 to 5: 1 know nothing to 5: know expert).
2. Have you developed Yahoo's Pipe Mashup Applications? Yes or No? If yes, how many Yahoo's Pipe Mashup applications have you developed?

Appendix 4 Pipes Assignment Survey

1. How many hours have you spent on learning Yahoo's Pipes (not counting working on the assignment)?
2. How many hours have you spent on the assignment?
3. In the following table, please rate the interestingness, usefulness and appropriateness of the assignment on the scale of 1 to 5, 1 being not interesting, 3 being neutral and 5 being most interesting, etc.

Assignment	Interestingness	Usefulness	Appropriateness
Part 1 Pipes Development and Report			
Part 2 Peer Evaluations			
Overall			

4. Now that you have completed the assignment, was the assignment more difficult or less difficult than you initially thought? (1: much more difficult; 3: about the same; 5 much easier).
5. What is the most difficult part of your assignment?

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096