Version Control for The Continuity of Learning Footprints Across Versions within E-Book Reader

Ching-Yuan Yang[†1] Brendan Flanagan[†2] Hiroaki Ogata[†2]

Abstract: This paper deals with the continuity of learning footprints across versions within an e-book reader, we propose the issue with e-book reading and data analysis after new version learning material been uploaded to the e-book reader. We first briefly give an introduction to talk about version control system and e-book reader, then we propose a process to combine the core concepts of version control such like *Diff* and *Merge* with an e-book reader, we also developed a *Transformation model* to keep the continuity of learning footprints across versions made by e-book users. In the final, we give the implementation results with our own test data about how this process can fix the proposed issue, and also the limitations, conclusions and future work of this research.

Keywords: Version Control, E-Book Reader, Learning Footprints, Continuity

1. Introduction

Version Control has become a critical part for taking control of the release history on various of learning fields, and meanwhile applied on software development and pedagogy in many researches. Version control can also reduce duplication of development effort, and simplified organization of shared materials [8]. In a coursework, problematic patterns of coding show up version control commit record and can provide clear feedback to software engineering students on how to improve their development processes [9]. In spite of this, we found that none of them mentioned how to keep the continuity of learning footprints made by e-book users across versions within e-book reader.

Due to the lack of the deployment on e-book based system, the primary objective of this research is to keep the continuity of learning footprints across versions within an e-book reader. Since this research propose a process to combine the core concepts of version control system such like *Diff* and *Merge* with an e-book reader to keep the continuity on the learning footprints across versions made by e-book users, there are some vocabulary and background need to be clarified in the first place, so we first give the introduction of version control system, e-book reader and also the issue we propose in the below.

1.1 Version control system

Version control, also known as revision control or source control, is the management of multiple versions of various information in a digit format. Such systems are commonly used to keep track of product evolution and backup in case of data loss [4]. Changes to these documents are identified by incrementing an associated number or letter code, termed the "revision number" or "revision" [1]. Each revision is associated with the person making the change. Revisions can be compared, restored, and in some cases, merged.

Usually a version control system will be organized in a client-server manner [4]. The server peer and client peer of a version control system are called *repository and working copy,* respectively. The task of the repository is to store historical data under version control system, and the working copy is pointed to the local copy from repository, each working copy contains only a single version number of the data to be changed.

A user interacts with a version control system by using a set of actions provided by the system. The following part of actions shows the common vocabulary and actions will be used in this research.

Repository - The *Repository* is where files' current and historical data are stored, often on a server. In this research, the *repository* is pointed to all the learning footprints logged in a local database.

Working copy - The *Working copy* is the local copy of files from a repository, and it contains only a single version number of the data to be changed. In this research, the *working copy* is pointed to the learning footprints that will be copied and modified from the repository.

Diff - Also called *Change,* represents a specific modification to a document under version control system. In this research, *Diff* indicate the modification of pages between two different versions of the same learning material which been uploaded to the e-book reader.

Merge - A *Merge* or *Integration* is an action in which two sets of modifications are applied to a file or set of files. In this research, *Merge* is deployed to merge the learning footprints from working copy after the modification by the proposed transformation model.

†1 Graduate School of Informatics, Kyoto University

†2 Academic Center for Computing and Media Studies, Kyoto University

1.2 E-book reader

E-book reader, also known as digital learning material reader, has become not only one of the most core part of modern formal education, but also an efficient data collection source in learning analytics area. The reading behavior made by students has previously been used to visualize class preparation and review patterns [2]. This research proposed the BookRoll digital learning material reading system. As shown in Figure 1, there are many types of learning footprints users can make, users can use marker function to highlight sections of learning materials in yellow for the sections that were not understood, or red for import sections. Memo function can also be created at any pages with the specific section of the page. Users can also use bookmark function to mark any pages or use full text search function to find the information they need. Currently, learning material contents can be uploaded to BookRoll in PDF format, and be able to support a large scale of devices as it can be accessed through a standard web browser [7].

For now, learning footprints made by users are being logged into a local database, there are all the information that be needed in the local database, include not only information about each version of learning materials but also all the learning footprints users made. In this research, the continuity of learning footprints across versions will be tracked and controlled properly within a local database. Due to the use of *Transformation model*, there are several reading behaviors in the BookRoll e-book reader need to be clarified. In the repository of BookRoll e-book reader, reading behaviors or action names will be stored in the repository, in Table 1 and Figure 1 we give the explanation of these reading behaviors, and the transformation model will be different between different types of reading behavior.

Table 1. Explanation of reading behaviors made by e-book users

Reading behavior	Explanation
NEXT	While a user click button “NEXT” and turn to the next page on e-book reader, the learning footprint will be saved as “NEXT”.
PREV	While a user click button “PREV” and turn to the previous page on e-book reader, the learning footprint will be saved as “PREV”.
PAGE_JUMP	While a user click button “jump” and jump to another specific page on e-book reader, the learning footprint will be saved as “PAGE_JUMP”.
MARKER	While a user click button “MARKER” to highlight the contents on e-book reader, the learning footprint will be saved as “ADD_MARKER”.
MEMO	While a user click button “MEMO” to take some notes on the specific page on e-book reader, the learning footprint will be saved as “ADD_MEMO”.
BOOKMARK	While a user click button “BOOKMARK” to mark the specific page on e-book reader, the learning footprint will be saved as “ADD_BOOKMARK”.

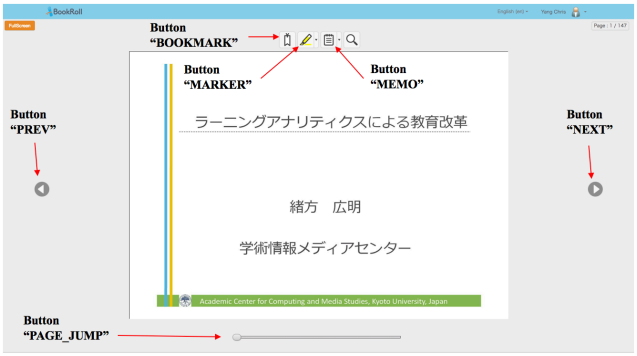

Figure 1. A screenshot of the E-Book Reader BookRoll

1.3 Illustration of version control issues in context of e-book reader

Usually many types of reading behaviors can be made by users on an e-book reader, include OPEN, NEXT, MEMO etc. In this research, in order to keep the continuity of these learning footprints made by users on the e-book reader, we propose a serious issue need to be addressed with.

First, these reading behaviors, or we call learning footprints on this research, need to be changed to the correct positions and page after a new version learning material been uploaded to the e-book reader, since several pages on the old version learning material might be changed or does not exist on the new version learning material, that will become a serious problem for the next reader who intend to read the same learning material. As shown in Figure 2, this two screenshots from the BookRoll e-book reader which will be mentioned in next section, indicate page 2 on the old version learning material and page 4 on the same learning material but with the new version. This two screenshots show the same page content but in the different page number of the same learning material, when the next reader intended to read the new version learning material, he or she can only see the learning footprints in the wrong page number, include memo, marker, bookmark made on the old version learning material because these learning footprints did not be changed to the correct positions and page numbers as shown in Figure 2. The exclamation mark shown in Figure 2 indicate that the positions or page numbers of the learning footprints made on the old version learning material might not be correct on the updated new version learning material.

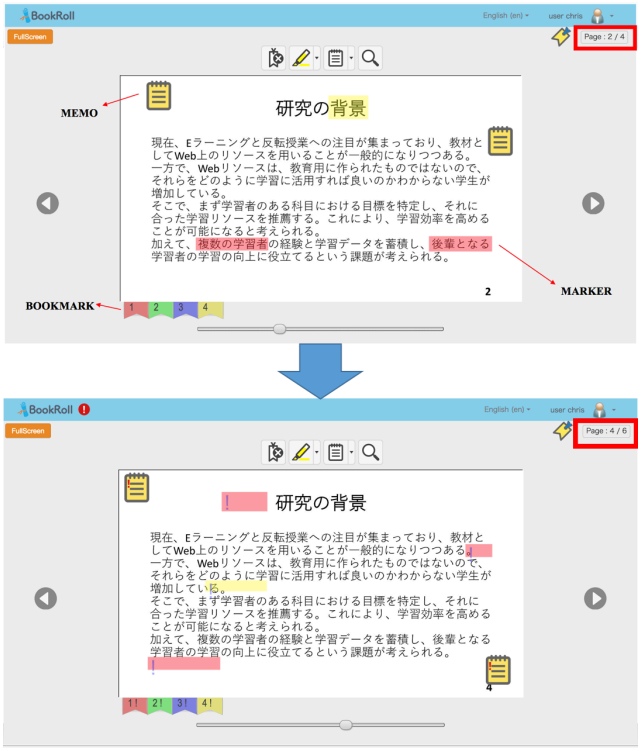

Figure 2. Two screenshots of page 2 on the old version learning material and page 4 on the updated new version learning material

Meanwhile, when we are trying to analyze these learning footprints that logged in a database, the result will go wrong if we don't modify the learning footprints with the old version learning material when a new version learning material been uploaded to the e-book reader and several pages on the old version learning material been changed on the new version learning material.

To solve the proposed issue above, we propose a process using *Diff* program, *Transformation Model* and *Merge* program which will be explained in Methodology section.

2. Related work

In this section, we give several related works talk about how researchers used version control system in the aspect of software development, coursework or project-based learning in the past. In [3], they introduced the population of a *Release History Database* that combines version and bug report data and further demonstrate some query examples with respect to software evolution analysis. In [1], they addressed with the development of a generalized model for version control systems application as a support in a range of project-based learning methods. In [5], they mentioned version control is not a peripheral or optional tool but a vital component in the software development toolkit, and also presented a model for incorporating version control into the entire project-based CS curriculum using a modern distributed version control system, Mercurial. In [4], they presented a method that continuously mines all known D-VCSs (Distributed Version Control Systems) of a software project to uncover the complete development history of a project. In [8], they reported on an exploratory project in which two instructors and an undergraduate teaching assistant used the Subversion version control system to collaborate remotely on developing and running two CS1 classes, and shared the results of using version control in CS1 to reduce administrative demands and to support creative collaboration between two instructors and an undergraduate teaching assistant. In [9], they also used the Subversion version control system over CVS to help make clear how a software engineering student developed programs and gave two figures as their results to emphasize that using version control did help make student development processes more visible, besides, they also mentioned that a brief scan over a student's commit logs showed whether the student was making incremental progress or waiting to work on the assignment until just before a deadline. In [10], they used an approach to teaching operating systems based on virtual appliances, a distributed version control system and live demonstrations, they combined a virtual appliance VMware with a distributed version control system Git to provide reliable storage for students' homework assignments, support students working together on group homework assignments, and manage the submission and grading of homework assignments, then they leveraged virtual appliances and the version control system to enable students to do live demonstrations of their work as part of grading their assignments to simplify grading, providing better feedback to students and greater interaction between instructional staff and students to facilitate learning.

Despite so many researches focused on software development, pedagogy, coursework or project-based learning, there is still a lack of the deployment on e-book based systems, so as the primary objective, this research will focus on version control for the continuity of learning footprints across versions within an e-book-based system. We propose a process include *Diff* program, *Transformation model* and *Merge* program in this paper, and the methodology of this research will be explained in the next section.

3. Methodology

In order to deal with the issue above, this research propose a process to combine the core concepts of version control with an e-book reader to keep the continuity of the learning footprints across versions obtained from e-book users, for now, we use some test data made by ourselves as our learning footprints for the propose process. As shown in Figure 3, after a new version learning material been uploaded to the BookRoll e-book reader, we will execute *Diff* program then be able to generate the *page control data* which will be shown in Figure 4(the modification of pages between two different versions of the same learning

material from the e-book reader). After that, all the learning footprints made by users on the old version learning material will be copied as a working copy, then we will use *Transformation model* to modify all the learning footprints in the working copy by following the proposed page control data, in the final of this process, these learning footprints will be added back to the main repository by deploying *Merge*.

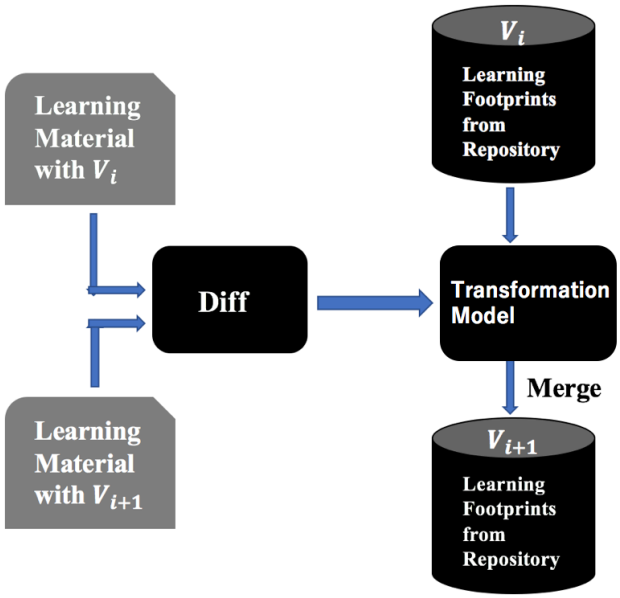

Figure 3. Process of the proposed method

3.1 Diff and page control data

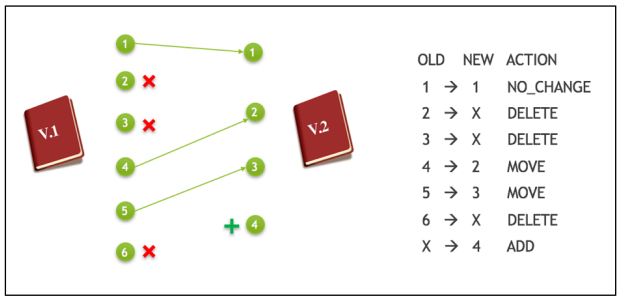

Figure 4. The page control data about how pages changed

As mentioned before, *Diff* program can be treated as the modification of pages between two different versions of the same learning material from the BookRoll e-book reader (*page control data*). This is the first part to be executed of this process. As shown in Figure 4, the picture in the left side indicates that there are version 1 and version 2 of the same learning materials with 6 pages and 4 pages, respectively and also shows how pages changed from version 1 to version 2 in image format. The picture in the middle indicates the page control data about how these same pages changed from version 1 to version 2, or the old version learning material to the new version learning material in text format, so according to the page control data, page 1, 4, 5 in version 1 learning material are ninety or one hundred percent similar to page 1, 2, 3 in version 2 learning material, respectively. By following the page control data shown in Figure 4, the rest parts can be executed properly in the next section.

3.2 Transformation model

Transformation model will be used to make a copy of these learning footprints and modify them in the working copy. This research proposes every type of transformation model as below. In the transformation model, we propose notations B and B' as the set of page contents from the old version learning material and the new version learning material, respectively, i and j are page numbers, 'm' and 'n' are real numbers, E and e indicate the set of learning footprint numbers and numbers of a stack of learning footprints made by one user in one e-book use. In this paper, B, B' and E, can be described as below:

$B = \{1,2,3,4,5,6\}$, $B' = \{1,2,3,4\}$, $E = \{1,2,3 \ldots e\}$.

In the proposed transformation model, we deploy three steps to execute the model, and they will be explained below.

3.2.1 Step 1

Step 1 of the proposed transformation model indicates the first transformation and the addition of flag.

First transformation indicates the transformation for the pages still exist after new version learning material been uploaded to the BookRoll e-book reader, which means a page is been detected that it belongs to NO_CHANGE or MOVE group from the page control data by *Diff* program as shown in Figure 4.

Addition of flag indicates that when we detected the page number of a learning footprint is not exist in new version learning material anymore and this learning footprint supposed to be removed from the working copy, in step 1 we will add a flag to it and then the learning footprint with flag will be deal with in step 2 or step 3, the specific method of this step will be shown below (in step 1 of the proposed transformation model, $\emptyset$ indicates that this learning footprint supposed to be removed from the working copy).

ADD/DELETE_MARKER *(ADD/DELETE_MEMO, ADD/DELETE_BOOKMARK, OPEN, CLOSE* will follow the same rule*)*:

- ***DIFF_ADD/DELETE_MARKER(B_i, B'_j)***

$$= \begin{cases} \emptyset, & if\ \{B_i \notin B'\} \\ \text{ADD/DELETE\_MARKER}(B'_j), & if\ \{B_i = B'_j\}; i \neq j \\ \text{ADD/DELETE\_MARKER}(B_i), & if\ \{B_i = B'_j\}; i = j \end{cases}$$

PAGE_JUMP *(BOOKMARK_JUMP and SEARCH_JUMP* will follow the same rule*)*:

- ***DIFF_PAGE_JUMP(B_i, B'_j, B_{i+m}, B'_{j+n})***

$$= \begin{cases} \emptyset, & if\ (\{B_i = B'_j\}\ \&\ \{B_{i+m} \notin B'\}) \mid \{B_i \notin B'\};\ m \neq 0 \\ PAGE\_JUMP(B'_j, B'_{j+n}), & if\ \{B_i = B'_j\}\ \&\ \{B_{i+m} = B'_{j+n}\}; i \neq j, m \neq 0, n \neq \{0, \pm 1\} \\ NEXT(B'_j), & if\ \{B_i = B'_j\}\ \&\ \{B_{i+m} = B'_{j+n}\}; m \neq 0, n = 1 \\ PREV(B'_j), & if\ \{B_i = B'_j\}\ \&\ \{B_{i+m} = B'_{j+n}\}; m \neq 0, n = -1 \\ PAGE\_JUMP(B_i, B_{i+m}), & if\ \{B_i = B'_j\}\ \&\ \{B_{i+m} = B'_{j+n}\}; i = j \&, m = n, \{m, n\} \neq \{0, \pm 1\} \\ OPEN(B'_{j+n}), & if\ DIFF\_OPEN(B_i, B'_j) = \emptyset\ \&\ \{B_i \notin B'\}\ \&\ \{B_{i+m} = B'_{j+n}\}; m \neq 0 \\ & \quad \&\ min(E) = e; e \in E \\ CLOSE(B'_j), & if\ DIFF\_CLOSE(B_{i+m}, B'_{j+n}) = \emptyset\ \&\ \{B_{i+m} \notin B'\}\ \&\ \{B_i = B'_j\}; m \neq 0 \\ & \quad \&\ \max(E) = e; e \in E \end{cases}$$

NEXT:

- ***DIFF_NEXT(B_i, B'_j, B_{i+1}, B'_{j+n})***

$$= \begin{cases} \emptyset, & if\ (\{B_i = B'_j\}\ \&\ \{B_{i+1} \notin B'\})\ |\ \{B_i \notin B'\} \\ \mathrm{NEXT}(B'_j), & if\ \{B_i = B'_j\}\ \&\ \{B_{i+1} = B'_{j+n}\}; i \neq j, n = 1 \\ \mathrm{NEXT}(B_i), & if\ \{B_i = B'_j\}\ \&\ \{B_{i+1} = B'_{j+n}\}; i = j, n = 1 \\ \mathrm{PREV}(B'_j), & if\ \{B_i = B'_j\}\ \&\ \{B_{i+1} = B'_{j+n}\}; n = -1 \\ PAGE\_\mathrm{JUMP}(B'_j, B'_{j+n}), & if\ \{B_i = B'_j\}\ \&\ \{B_{i+1} = B'_{j+n}\}; n \neq \{0, \pm 1\} \\ \mathrm{OPEN}(B'_{j+n}), & if\ DIFF\_OPEN(B_i, B'_j) = \emptyset\ \&\ \{B_i \notin B'\}\ \&\ \{B_{i+1} = B'_{j+n}\} \\ & \quad \&\ min(E) = e; e \in E \\ \mathrm{CLOSE}(B'_j), & if\ DIFF\_CLOSE(B_{i+1}, B'_{j+n}) = \emptyset\ \&\ \{B_{i+1} \notin B'\}\ \&\ \{B_i = B'_j\} \\ & \quad \&\ \max(E) = e; e \in E \end{cases}$$

PREV:

- ***DIFF_PREV(B_i, B'_j, B_{i-1}, B'_{j+n})***

$$= \begin{cases} \emptyset, & if\ (\{B_i = B'_j\}\ \&\ \{B_{i-1} \notin B'\})\ |\ \{B_i \notin B'\} \\ \mathrm{PREV}(B'_j), & if\ \{B_i = B'_j\}\ \&\ \{B_{i-1} = B'_{j+n}\}; i \neq j, n = -1 \\ \mathrm{PREV}(B_i), & if\ \{B_i = B'_j\}\ \&\ \{B_{i-1} = B'_{j+n}\}; i = j, n = -1 \\ \mathrm{NEXT}(B'_j), & if\ \{B_i = B'_j\}\ \&\ \{B_{i-1} = B'_{j+n}\}; n = 1 \\ PAGE\_\mathrm{JUMP}(B'_j, B'_{j+n}), & if\ \{B_i = B'_j\}\ \&\ \{B_{i-1} = B'_{j+n}\}; n \neq \{0, \pm 1\} \\ \mathrm{OPEN}(B'_{j+n}), & if\ DIFF\_OPEN(B_i, B'_j) = \emptyset\ \&\ \{B_i \notin B'\}\ \&\ \{B_{i-1} = B'_{j+n}\} \\ & \quad \&\ min(E) = e; e \in E \\ \mathrm{CLOSE}(B'_j), & if\ DIFF\_CLOSE(B_{i-1}, B'_{j+n}) = \emptyset\ \&\ \{B_{i-1} \notin B'\}\ \&\ \{B_i = B'_j\} \\ & \quad \&\ max(E) = e; e \in E \end{cases}$$

To specify the last two conditions of the transformation model for learning footprints with reading behavior *PAGE_JUMP, NEXT* and *PREV,* all the learning footprints made by one user in one e-book can be logged like a stack start from a learning footprint with reading behavior *OPEN* and end of a learning footprint with reading behavior *CLOSE*. For instance, we have E as the set of learning footprint numbers and e as the numbers of a stack of learning footprints made by one user in one e-book use. In the transformation model, min(E) and max(E) indicate the minimum and maximum number e of set E, respectively.

3.2.2 Step 2

Step 2 of the proposed transformation model indicates the purge of part of the learning footprint with flag by using the developed algorithm shown in Figure 5, if both the initial page and the terminal page of learning footprint number e are not exist in the new version learning material, this learning footprint will be removed from the working copy by the proposed purge algorithm (to demonstrate *PI* and *PT*, take learning footprints with reading behavior *NEXT* and page number 3 as an example, *PI* and *PT* indicate page 3 and page 4 in this learning footprint, respectively as shown in Table 2).

```
algorithm Purge()
    for each e in E do
        if FLAG_NEXT(E_e) = TRUE and EXIST_PI_e = FALSE and EXIST_PT_e = FALSE
            DEL()
        if FLAG_PREV(E_e) = TRUE and EXIST_PI_e = FALSE and EXIST_PT_e = FALSE
            DEL()
        if FLAG_PAGE_JUMP(E_e) = TRUE and EXIST_PI_e = FALSE and EXIST_PT_e = FALSE
            DEL()
```

Figure 5. The developed algorithm for the purge

3.2.3 Step 3

Step 3 of the proposed transformation model indicates the second transformation of the rest learning footprints with flag.

For learning footprints with flag and reading behavior *PAGE_JUMP*, *NEXT* or *PREV*, we developed another algorithm to keep the continuity of these learning footprints as shown in Figure 6 (here we give the reading behavior *PAGE_JUMP* as an example, reading behavior *NEXT* and *PREV* will actually follow the same algorithm), we also give Table 2 to show the definition of all the variables which will be applied in the developed algorithm for Step 2 and Step 3.

```
algorithm Second_Transformation()
    for each e in E do
        if FLAG_PAGE_JUMP(E_e) = TRUE and FLAG_RB(E_e+1) = TRUE and (PT_e+1 - PI_e) = -1
            return PREV(PI_e)
        if FLAG_PAGE_JUMP(E_e) = TRUE and FLAG_RB(E_e+1) = TRUE and (PT_e+1 - PI_e) = 1
            return NEXT(PI_e)
        if FLAG_PAGE_JUMP(E_e) = TRUE and FLAG_RB(E_e+1) = TRUE and |(PT_e+1 - PI_e)| > 1
            return PAGE_JUMP(PI_e, PT_e+1)
```

Figure 6. The developed algorithm for the second transformation with reading behavior PAGE_JUMP

Table 2. Definition of variables applied in the developed algorithm for Step 2 and Step 3

Variable	Definition
$FLAG\_PAGE\_JUMP(E_e)$	The flag of learning footprint number e with reading behavior *PAGE_JUMP*. (TRUE indicates with flag, FALSE indicates without flag, and reading behavior *NEXT* and *PREV* will follow the same definition)
$FLAG\_RB(E_{e+1})$	The flag of learning footprint number **e+1** with any type of reading behavior.
PT_{e+1}	The terminal page of learning footprint number **e**.
PI_e	The initial page of learning footprint number **e**.
$PREV(PI_e)$	A learning footprint with reading behavior *PREV*, and the initial page of learning footprint number **e**.
$NEXT(PI_e)$	A learning footprint with reading behavior *NEXT*, and the terminal page of learning footprint number **e**.
$PAGE\_JUMP(PI_e, PT_{e+1})$	A learning footprint with reading behavior *PAGE_JUMP*, and a jump from the initial page of learning footprint number **e** to the terminal page of learning footprint number **e+1**.
$EXIST\_PI_e$	The existence in new version learning material of the initial page of learning footprint number **e**. (TRUE indicates exist, FALSE indicates not exist, and $EXIST\_PT_e$ will follow the same definition)
DEL()	An action to remove that learning footprint.

3.3 Merge

Merge will be executed after all the learning footprints in the working copy been modified by using transformation model. This working copy will be added back to the original repository again and become learning footprints for the current version learning material by deploying *Merge*.

As mentioned in the previous sections, every learning footprint made by users on the interface of the BookRoll e-book reader need to be modified to the correct position once a new version learning material been uploaded to the e-book reader. By deploying the proposed *Diff, Transformation model and Merge* program, all the learning footprints made by users on the old version learning material can be modified properly to the correct position on the new version learning material, include updated or removed, and the implementation results of this section will be summarized below.

4. Implementation

This research implemented the core concepts of version control to modify the learning footprints made on the BookRoll e-book reader by users, and also create and modify the working copy with the learning footprints logged into the repository of the BookRoll e-book reader. The proposed issue can be fixed after the process proposed by this research.

As shown in Figure 7, all the learning footprints made by users, include *MEMO, MARKER* and *BOOKMARK* on the interface of the BookRoll e-book reader will be changed to the correct

positions and page numbers after deploying the proposed process.

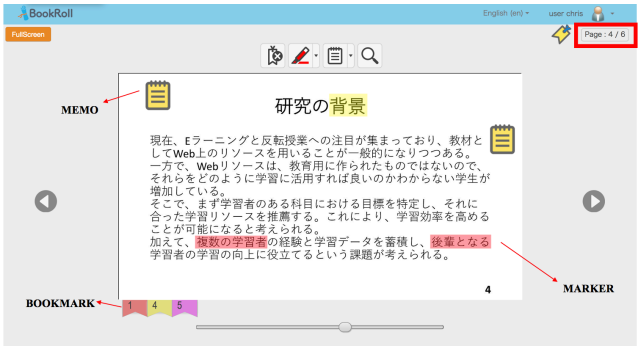

Figure 7. Page 4 on the new version learning material after the propose process

As shown in Figure 8-12, we propose E and e as the set of learning footprint numbers and numbers of a stack of learning footprints made by one user in one e-book use.

In step 1 of transformation model, E_1 will be remained as the same in the working copy, E_2 will be transformed to NEXT(B_2'), E_3 will be transformed to ADD_MEMO(B_2'), E_4, E_5 and E_6 will be added a flag, E_7 will be transformed to PAGE_JUMP(B_1', B_3'), E_8 will be transformed to ADD_MARKER(B_3'), E_9 will be transformed to CLOSE(B_3'), E_{10} will be removed from the working copy as shown in Figure 9.

In step 2 of transformation model, there are two flag learning footprints, because E_5 matches *"EXIST_PI$_e$ = FALSE* **and** *EXIST_PT$_e$ = FALSE"* in the proposed Purge algorithm, so for this sample stack of learning footprints, E_5 will be removed from the working copy as shown in Figure 10.

In step 3 of transformation model, E_4 will be transformed to PREV(B_2') without flag and meanwhile, E_6 will be removed from the working copy as shown in Figure 11, finally, the implementation result to the stack of learning footprints in the repository and the page transition flow of this sample stack are shown in Figure 12 and Figure 13, respectively.

After deploying *Transformation model*, these learning footprints made by users in the working copy will be merged back to the main repository again as the learning footprints for version 2 learning material, and also be able to be used for the future data analysis.

E	Flag	Learning Footprints	Page Number	Date
1		OPEN	1	XX-XX-10-00-00
2		PAGE_JUMP	1 -> 4	XX-XX-10-00-15
3		ADD MEMO	4	XX-XX-10-00-20
4		PREV	4	XX-XX-10-00-40
5		PREV	3	XX-XX-10-01-05
6		PREV	2	XX-XX-10-01-35
7		PAGE_JUMP	1 -> 5	XX-XX-10-01-55
8		ADD_MARKER	5	XX-XX-10-02-20
9		NEXT	5	XX-XX-10-02-45
10		CLOSE	6	XX-XX-10-03-00

Figure 8. A sample stack of learning footprints made by users in the repository

E	Flag	Learning Footprints	Page Number	Date
1		OPEN	1	XX-XX-10-00-00
2		NEXT	1	XX-XX-10-00-15
3		ADD MEMO	2	XX-XX-10-00-20
4	1	PREV	4	XX-XX-10-00-40
5	1	PREV	3	XX-XX-10-01-05
6	1	PREV	2	XX-XX-10-01-35
7		PAGE_JUMP	1 -> 3	XX-XX-10-01-55
8		ADD_MARKER	3	XX-XX-10-02-20
9		CLOSE	3	XX-XX-10-02-45
10		CLOSE	6	XX-XX-10-03-00

Figure 9. The implementation to Step 1 of transformation model

E	Flag	Learning Footprints	Page Number	Date
1		OPEN	1	XX-XX-10-00-00
2		NEXT	1	XX-XX-10-00-15
3		ADD MEMO	2	XX-XX-10-00-20
4	1	PREV	4	XX-XX-10-00-40
5	1	PREV	3	XX-XX-10-01-05
6	1	PREV	2	XX-XX-10-01-35
7		PAGE_JUMP	1 -> 3	XX-XX-10-01-55
8		ADD_MARKER	3	XX-XX-10-02-20
9		CLOSE	3	XX-XX-10-02-45
10		CLOSE	6	XX-XX-10-03-00

Figure 10. The implementation to Step 2 of transformation model

E	Flag	Learning Footprints	Page Number	Date
1		OPEN	1	XX-XX-10-00-00
2		NEXT	1	XX-XX-10-00-15
3		ADD MEMO	2	XX-XX-10-00-20
4		PREV	2	XX-XX-10-00-40
5	1	PREV	3	XX-XX-10-01-05
6	1	PREV	2	XX-XX-10-01-35
7		PAGE_JUMP	1 -> 3	XX-XX-10-01-55
8		ADD_MARKER	3	XX-XX-10-02-20
9		CLOSE	3	XX-XX-10-02-45
10		CLOSE	6	XX-XX-10-03-00

Figure 11. The implementation to Step 3 of transformation model

E	Flag	Learning Footprints	Page Number	Date
1		OPEN	1	XX-XX-10-00-00
2		NEXT	1	XX-XX-10-00-15
3		ADD MEMO	2	XX-XX-10-00-20
4		PREV	2	XX-XX-10-00-40
7		PAGE_JUMP	1 -> 3	XX-XX-10-01-55
8		ADD_MARKER	3	XX-XX-10-02-20
9		CLOSE	3	XX-XX-10-02-45

Figure 12. The implementation result to the stack of learning footprints in the repository

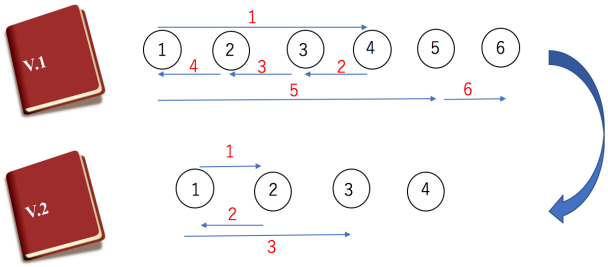

Figure 13. The page transition flow of this stack by the proposed pcrocess

5. Limitations

Currently this implementation will transform the learning footprints both on the interface of e-book system and also the main repository, so it is mainly benefit for e-book users, for researchers or administrators it will become another analysis issue since after the proposed process, all the learning footprints made by users will become different form the previous learning footprints which makes them hard to be analyzed as before. For example, *PAGE_JUMP* will be transformed to *NEXT* and *NEXT* will be transformed to *CLOSE* etc.

Meanwhile this implementation use *Diff* program to determine whether the two images are ninety or one hundred percent similar between two different versions of the same learning material, so the determination depend on every image of that learning material, but in many cases, the two images might really similar to each other, but they will still be determined as different images due to the proposed *Diff* program, which makes more development is needed in the proposed *Diff* program to make sure we can track any kind of different similarity of each two images.

Also, this implementation focused on the continuity of page numbers across versions to make these learning footprints and the interface of an e-book reader easy to analyze or read, respectively. But unfortunately, every time we remove some learning footprints by using transformation model, the time stamp of a specific stack of learning footprints in the main repository might go wrong because users might not actually do that action at that time since all the leaning footprints will be transformed by deploying the proposed process, and it actually needs to be well connected across versions within any stack of learning footprints.

6. Conclusions and future work

In this paper, a process to combine the core concepts of version control with an e-book reader to keep the continuity on the learning footprints across versions obtained from e-book users is presented. As shown in Figure 7-12, the proposed issue can be fixed by deploying the proposed process, also it indicates that version control for the continuity of learning footprints is necessary and be able to play a huge part for an e-book based system since instructor may update their learning material in anytime. For now, we just use our test data made by ourselves as our learning footprints, in the future we will use the real learning footprints made by other user, mainly students in a course, as our data sets.

Meanwhile, this implementation focused on ninety of one hundred percent similar images between two different versions of the same learning material, so the development to determine any kind of different similarity of each two images is ongoing.

Also, this implementation focused on the continuity of page numbers across versions to make sure these learning footprints can be connected properly in the main repository, in the future, we will also dedicate on the continuity of time stamp and try to make the time stamp of learning footprints connected properly across versions so that we can try to avoid any possible analysis issue for researchers or administrators after we update or remove some original learning footprints from the main repository..

References

[1] Milentijevic, Ivan, Vladimir Ciric, and Oliver Vojinovic. "Version control in project-based learning." *Computers & Education* 50.4 (2008): 1331-1338.

[2] Ogata, Hiroaki, et al. "Learning Analytics for E-Book-Based Educational Big Data in Higher Education." *Smart Sensors at the IoT Frontier*. Springer International Publishing, 2017. 327-350.

[3] Fischer, Michael, Martin Pinzger, and Harald Gall. "Populating a release history database from version control and bug tracking systems." *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on*. IEEE, 2003.

[4] German, Daniel M., Bram Adams, and Ahmed E. Hassan. "Continuously mining distributed version control systems: an empirical study of how Linux uses Git." *Empirical Software Engineering* 21.1 (2016): 260-299.

[5] Rocco, Daniel, and Will Lloyd. "Distributed version control in the classroom." *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 2011.

[6] Cederqvist, Per, et al. "Version management with CVS, 1992." *URL http://www. cvshome. org/docs/manual* (2006).

[7] FLANAGAN, Brendan, and Hiroaki OGATA. "Integration of Learning Analytics Research and Production Systems While Protecting Privacy."

[8] Clifton, Curtis, Lisa C. Kaczmarczyk, and Michael Mrozek. "Subverting the fundamentals sequence: using version control to enhance course management." *ACM SIGCSE Bulletin*. Vol. 39. No. 1. ACM, 2007.

[9] Glassy, Louis. "Using version control to observe student software development processes." *Journal of Computing Sciences in Colleges* 21.3 (2006): 99-106.

[10] Laadan, Oren, Jason Nieh, and Nicolas Viennot. "Teaching operating systems using virtual appliances and distributed version control." *Proceedings of the 41st ACM technical symposium on Computer science education*. ACM, 2010.