Lab 2 - Blackboard Archive Extractor

Team Crystal

Grant Atkins

CS411

Professor Thomas Kennedy

April 4, 2017

Version 1.2

Contents

List of Figures

1 Introduction

Blackboard Extractor is a software based application that is designed to reduce the manual effort involved in making past Blackboard content available again, separate from Blackboard. This software provides users with an interactive interface to select an archived Blackboard course archive from their computer. The user selected archived content will then be processed by this software and create a website based on the archived content from Blackboard.

Teaching professionals rely on Blackboard to display their course content. Blackboard does not offer a viable option to export this content for independent use. A Blackboard course archive can exceed five-hundred files in a directory making it difficult for teaching professionals to rebuild websites with this content. Teaching professionals currently have to: find an unused Blackboard course into which he or she can load the archive, manually download course content page-by-page, manually format folder structure for content, and then repeat the previous process for each archive. The process is noted to be completely manual as shown in Figure 1.

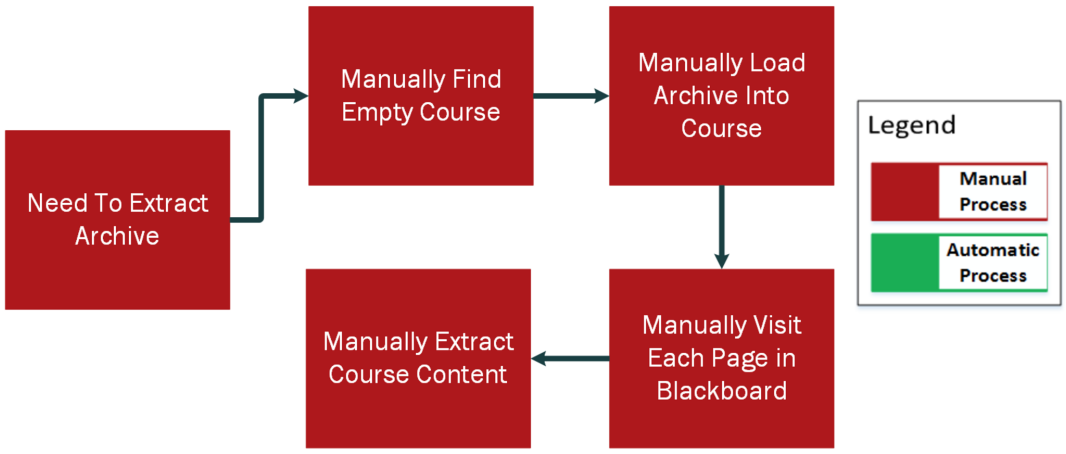

Figure 1. Current process flow of Blackboard extraction

(This space is intentionally left blank.)

1.1 Purpose

Blackboard Extractor aims to reduce the effort involved to create a website based on a Blackboard Course Archive. As shown in Figure 2, the process is purely manual and the Blackboard Extractor intends to solve this by effectively reducing the time involved in extraction. Blackboard Extractor is designed to process content automatically, link relative files automatically, and extract a website for teaching professionals to display their archived Blackboard courses.

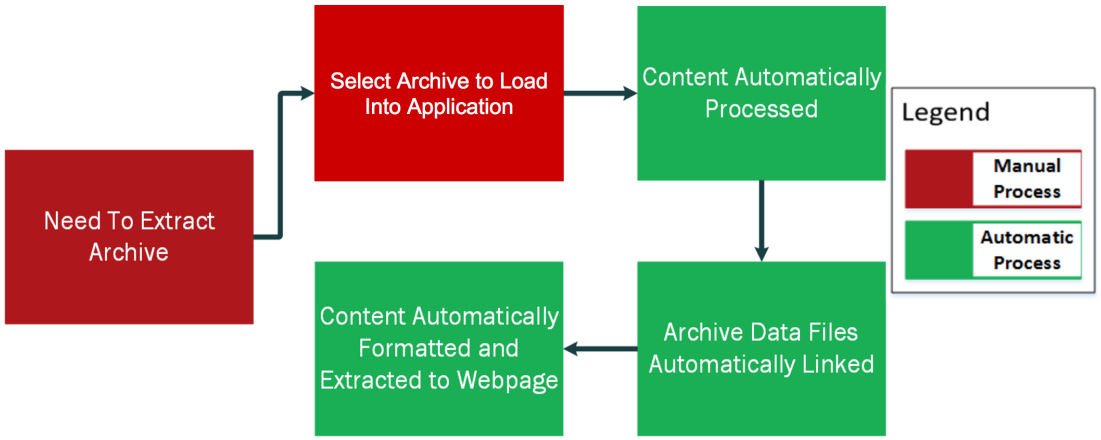

Figure 2. Process flow of Blackboard extraction with Automatic processes

1.2 Scope

According to the Old Dominion University Faculty Handbook, lecturers are reviewed every year, senior lecturers are reviewed every three years, and tenured professors are reviewed every five years (*ODU Faculty Handbook*, 2005). Professors are required to present all materials used throughout all courses taught since the previous review, including any material stored on Blackboard. Blackboard only stores courses for two years, before which the course must be exported as a Blackboard Archive file. If a Blackboard archive is not exported, the content on Blackboard will be lost after two years making. Blackboard Extractor aims to export the content, provided from the Blackboard archive, for ODU professors and other teaching professionals.

1.3 Definitions, Acronyms, and Abbreviations

GUI: Graphical User Interface

IMS Course Cartridge: A standardized course cartridge format that enables a way of packaging and exchanging digital materials

Blackboard Archive: Course content that was saved by Blackboard

Microsoft .NET: Programming infrastructure for building, deploying, and running applications that use .NET technology such as desktop applications

Root Index: Describes the first position of a collection of data elements in a structure

UI: User Interface

UX: User Experience

HTML: HyperText Markup Language which forms basic building block of web by defining content of a webpage

1.4 References

Atkins, G. (2017, February 6). *Lab 1 - blackboard extractor.*

Corcoran, B. (2014, July 23). *Blackboard's jay bhatt strikes up the brass band.* Retrieved from `https://www.edsurge.com/news/2014-07-23-blackboard-s-jay-bhatt-strikes-up-the-brass-band`

Odu faculty handbook. (2005, December 21). Retrieved from `https://www.odu.edu/content/dam/odu/offices/human-resources/docs/faculty-handbook.pdf`

1.5 Overview

This product specification provides the hardware and software configuration and capabilities and features to fully implement the Blackboard Extractor prototype. This prototype will not need the use of any external interfaces. The information provided in the remaining sections of this document includes a detailed description of the hardware and software configuration and capabilities and features. The product specification requirements provided in Lab 2 Section 3.1 can be found in a separate document.

2 General Description

Blackboard Extractor will consist of single software solution usable through either command line interface or a GUI. The end user will provide Blackboard archive to the Blackboard Extractor. After specifying an output location, the Blackboard Extractor will process all the files from the Blackboard Archive ultimately creating a directory of files that form a website representing that material (Atkins, 2017).

2.1 Prototype Architecture Description

The Blackboard Extractor prototype has three main features, which include: a user interface, algorithms for automation, and a website as an end result.

The graphical user interface (GUI) will allow for a user to select a Blackboard archive zip file from their local machine. Users will also have to specify a location to save the resulting web content created by the Blackboard Extractor. When both fields for input and output have been filled, the submit button will be made available to press and execute the Blackboard Extractor program much like in Figure 3. Any errors that show in during or after the processing of a Blackboard archive will be represented through this GUI. Any errors that occur using the command line interface (CLI) will be reported through this interface.

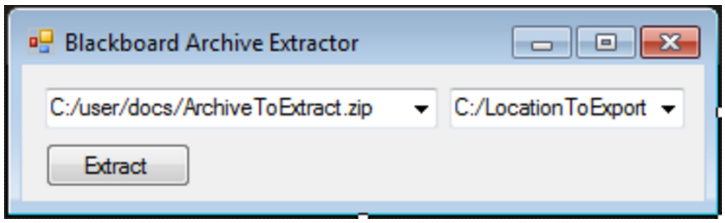

Figure 3. GUI Example

Blackboard Extractor also has an option to be executed from CLI. The command line interface will take two parameters after the program name: input location and output location. Input location being the location of the Blackboard archive the user wants to provide, and the output location being where the created files want to be saved very similar to Listing 1.

```
blackboardExtractor.exe ./inputFiles.zip ./outputDirectory
```

Listing 1: Sample of command line execution

There are four major interactions with the Blackboard Extractor: user interaction with the GUI, content parsing, content linking, and finally content reformatting. The Blackboard Extractor program is only responsible for all of these interactions, with no requirements from outside sources except for an internet connection to check if URIs found are still alive. As shown in Figure 4, a user provides archive content which follows the steps of parsing, linking and reformatting ultimately creating a website from the content.

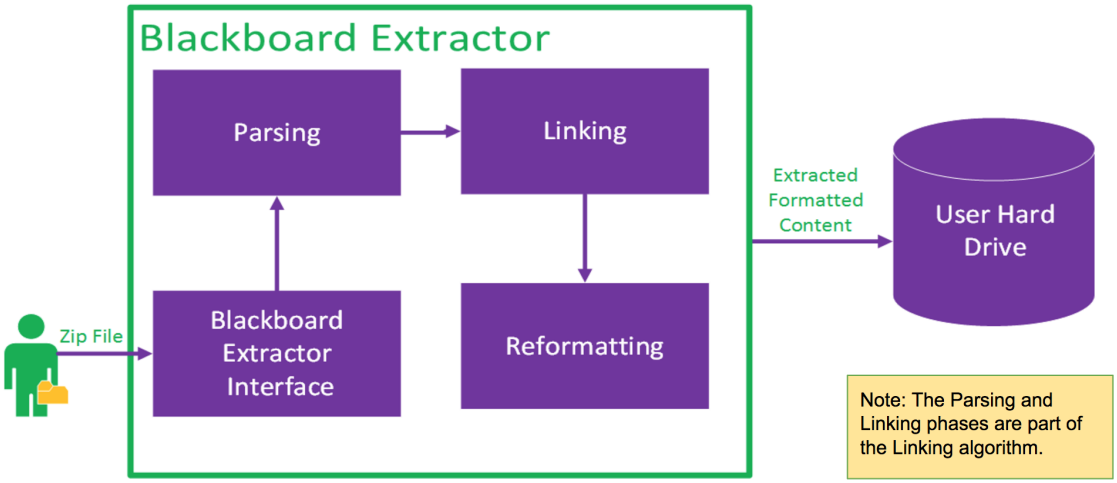

Figure 4. Major Functional Components Diagram

Parsing is part of the linking phase algorithm, performing a very large task creating model objects that represent various files and hyperlinks of a Blackboard archive. The parsing phase must discover the manifest file inside of Blackboard course archive before the linking starts. The parsing component is also responsible for receiving a zip file and unpacking it. It will then iterate through all the resource files containing course content for the Blackboard archive and give these files to the next phase, linking.

2.2 Prototype Functional Description

Blackboard Extractor aims to reduce the manual time by incorporating two main algorithms: linking and formatting. Before, teaching professionals would have to manually sort through the content of an archive, find which files reference other files, and then finally format all the content in these archives. Sorting through the content and then linking other content will be handled by the linking phase algorithm. Creating a directory and then converting the content to valid HTML will be handled by the formatting algorithm. This will automate the process to format and create these kinds of websites. A user will still have to find a Blackboard Archive file for this program and select an output destination for the website, but this program will automate everything else.

Blackboard provides an archive format that provides referential links to each of the files within the archive. This is taken from the *manifest.xml* file found within the archive. The manifest file can define the course name, main sections, sub sections, date, reference other content, and much more. In Figure 5, there is an example of the Blackboard Archive pointing to blog posts, homework assignments, and group discussions which were all past content from a course. If the *manifest.xml* file is not provided then this program will be unable to determine relationships between files in the archive.

The default structure of a Blackboard course archive is modeled for the purpose of processing. This allows for relationships between topics and content to be established through the use of an identifier for a resource's content. The linking phase of the major components aims to recreate the relationships of an Blackboard archive by establishing parent and child relationships. At the top level, a parent is a topic and the children of this parent will be the resource content under this topic. Children nodes that link with more content by hyperlinks can also be considered parents. To ensure that content isn't repeatedly referenced by different nodes, parent nodes will keep track of which resources it has access to by keeping a cache. This process will repeat until the last child is found as depicted in Figure 6. If the last child is found in the manifest file, the linking will

then pass the resulting tree structure, for example as shown in Figure 7, to the reformatting component.

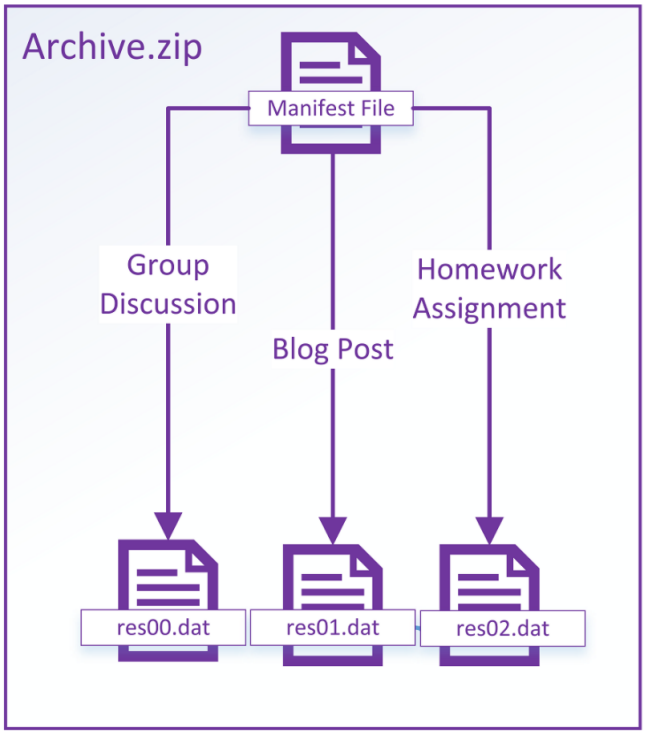

Figure 5. Blackboard archive format and linking examples

(This space is intentionally left blank.)

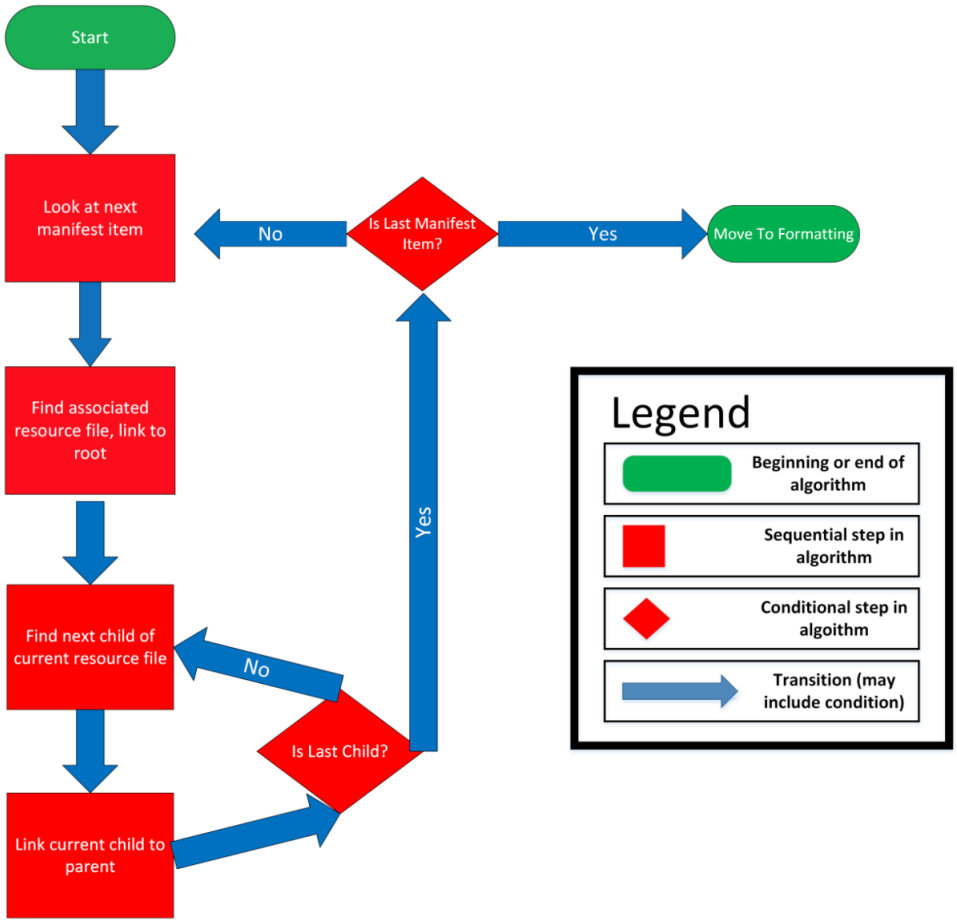

Figure 6. Linking phase algorithm

Reformatting phase is when the linking phase has given it a new tree structure and its ready to form a new file structure. When reformatting, the program will iterate through each node of the tree first converting it to HTML if it is an HTML document. If it is not an HTML document but rather a pdf or an image, these will placed in a separate directory based on their file type. When the algorithm has gone through each node it will finish creating the website directory and save the contents to the user's local hard drive.

(This space is intentionally left blank.)

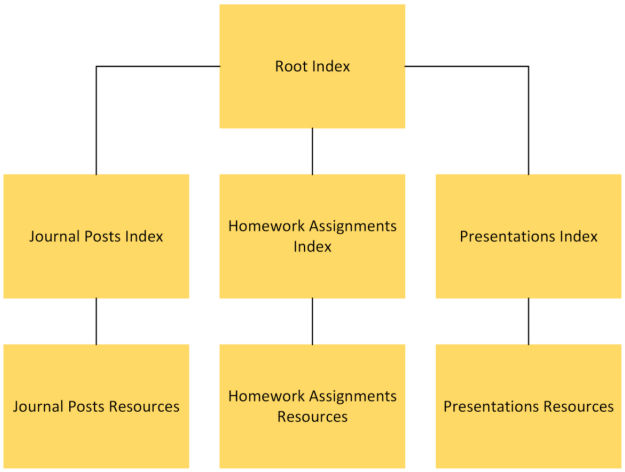

Figure 7. Example of linking phase resulting tree structure

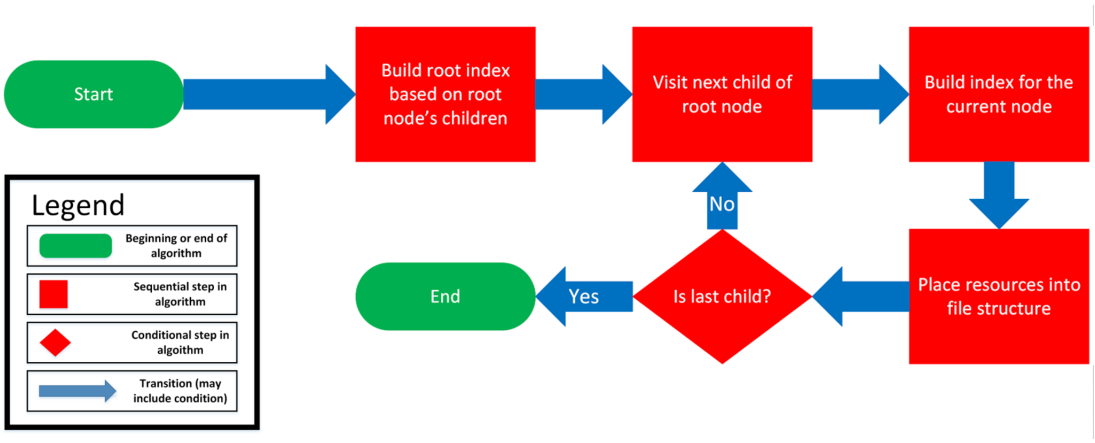

Figure 8. Reformatting phase algorthim

2.3 External Interfaces

No external Interfaces will be needed to construct the Blackboard Extractor prototype. No additional resources are required to be purchased for the development and demonstration of the Blackboard Extractor prototype.

3 Specific Requirements

The functional requirements of the Blackboard Extractor prototype, found in section 3.1, are located in a separate document. Lab 2 Section 3.1 contains all requirements that are necessary to complete the prototype.