Lab 1 - Blackboard Extractor

Author:
Grant Atkins

Instructor:
Thomas Kennedy

February 6, 2017

Contents

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 from their computer. The archived content that the users select 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 class content but Blackboard doesn't 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 professional 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 they have. The process is by no means automatic as illustrated in Figure 1.

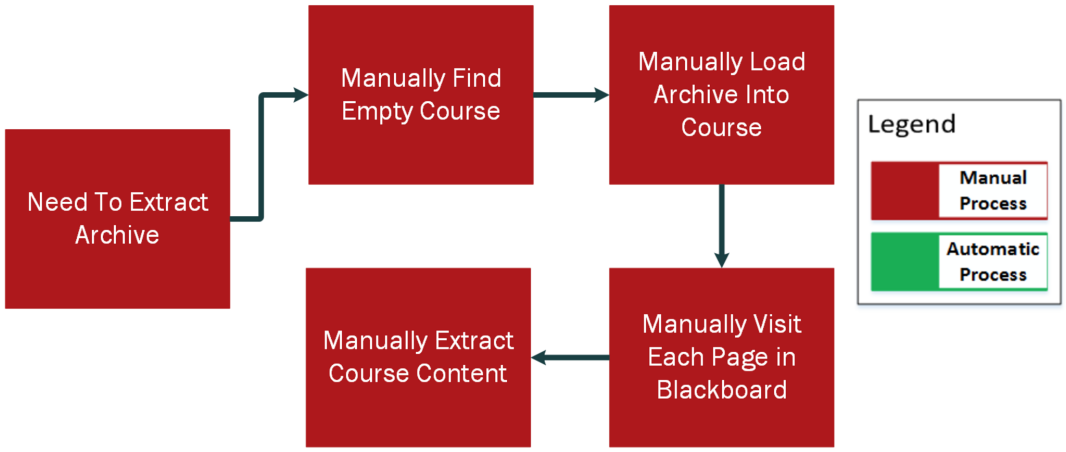

Figure 1: Current process flow of Blackboard extraction

1.1 Blackboard in Learning Institutions

Blackboard is an online tool used by schools, colleges, and universities to organize the course materials of a class for both professors and students. It allows

professors to provide information such as syllabi and homework assignments to students over the Internet. Blackboard offers the option of exporting previously used classes as an archives, which offers teaching professionals the option to reuse this content in Blackboard.

1.2 ODU Faculty Review Process

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 [4]. 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.

1.3 Solution Overview

Blackboard Extractor will significantly reduce the time that teaching professionals have when processing archived Blackboard content. Our solution will be an application which allows a user to load a Blackboard course archive and explore the contents of the archive. End users will rely on this application to automatically process the content of Blackboard archives instead of the current manual process of individually traversing through a Blackboard course content archive illustrated by the new process flow in Figure 3. The speed of this application will depend on the amount of content a Blackboard course cartridge holds and the user's internet connection.

2 Blackboard Extractor Product Description

Blackboard Extractor will consist of single software solution available from command line interface or a GUI. The end user will provide the Blackboard archive files to the Blackboard Extractor. After specifying an output location for the files, 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.

2.1 Product Features

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

2.1.1 User Interface

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 created from the program. 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 2. Any errors that occur during or after the processing of a Blackboard archive will be represented through this GUI.

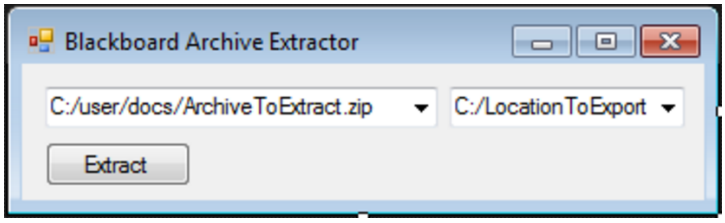

Figure 2: Example

Blackboard Extractor also has an option to be executed from command line interface (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

2.1.2 Algorithms and Process Flow

The 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 drastically reduce the amount of time spent formatting and creating these kinds of websites. A user will still have to find the files to use for this program but this program will automate everything else.

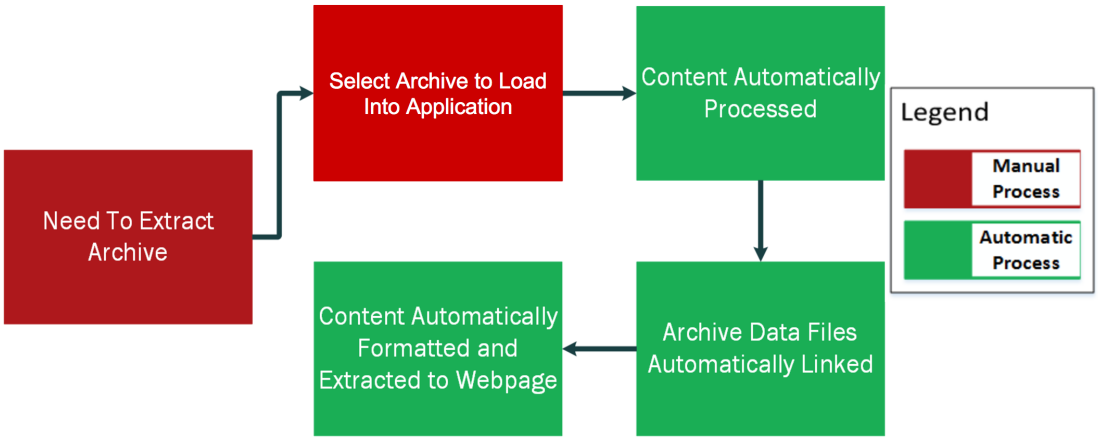

Figure 3: Current process flow of Blackboard extraction

2.1.3 Output End Result

The end goal of the Blackboard Extractor is to create a website from the content provided in an archived zip file. The website will provide a sidebar to access the main sections of the content on the left. In the center there will be content from each of the current selected section. To ease users the website format will be very similar to blackboard's current layout as shown in Figure 4.

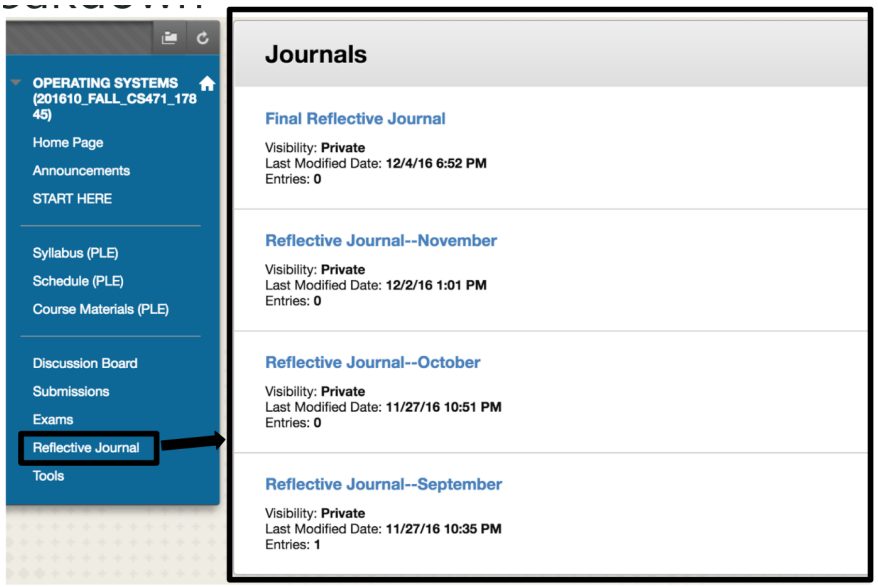

Figure 4: Blackboard website with a side bar and content

2.2 Major Components

There really is four major interactions with the Blackboard Extractor: user interaction with the GUI, parsing through the content, linking together all the content, and finally reformatting all the content. The Blackboard Extractor program is 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.

2.2.1 MFCD

As shown in Figure 5, the user passes in a zipped Blackboard archive through the GUI. The GUI then gives the zip file to the parsing component. Parsing then leads to linking of the files. Finally reformatting will create the resulting file structure and send it to the output location.

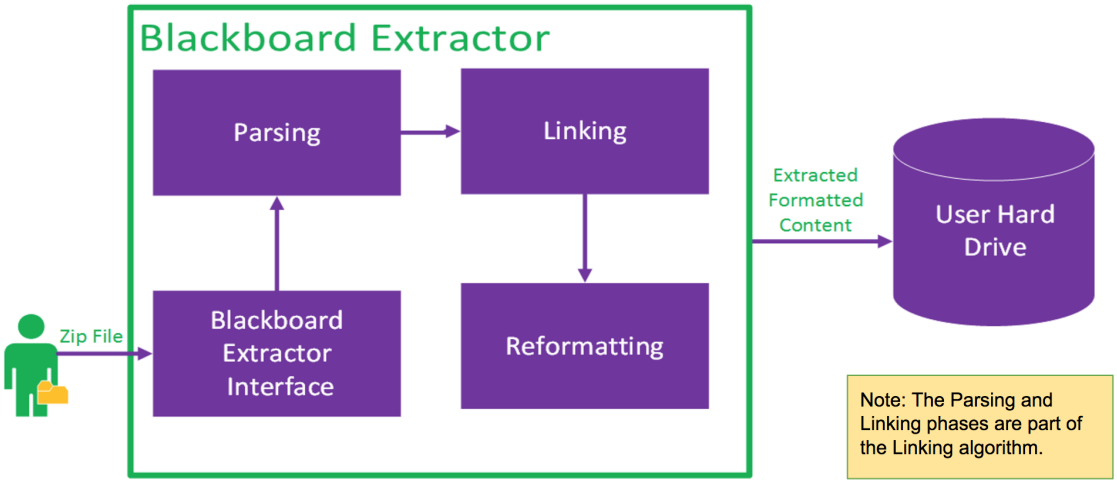

Figure 5: Major Functional Components Diagram

2.2.2 Parsing

Parsing is really part of the linking phase algorithm, but it deserves its own component because it performs a very large task before the linking starts. The parsing component is responsible for receiving a zip file and unpacking it. It will then iterate through all the files and give these files to the next phase, linking.

2.2.3 Linking

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 **??**, 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 Archive is a graph, meaning that there are files within the Blackboard archive that can point back back and forth to other files. The linking phase of the major components aims to create a new tree

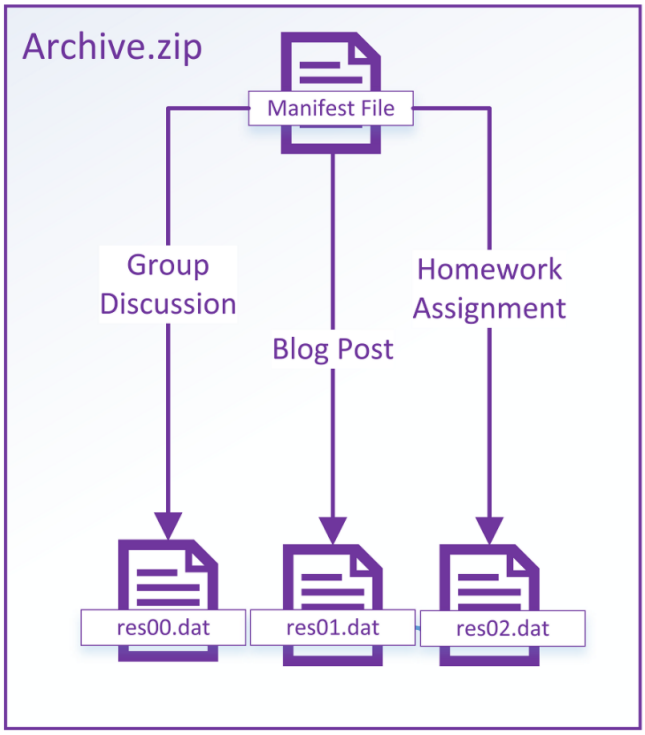

Figure 6: Blackboard archive format and linking examples

data structure with parent and child and parent relationships. To establish this relationship, the manifest will be used to establish parent relationships and then children will be created by the reference from parent to child files. To prevent this from conforming back into a graph each parent will keep a cache of files already as its children, so the same files won't be referenced more than once. This process will repeat until the last child is found as depicted in Figure 7.. 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 8, to the reformatting component.

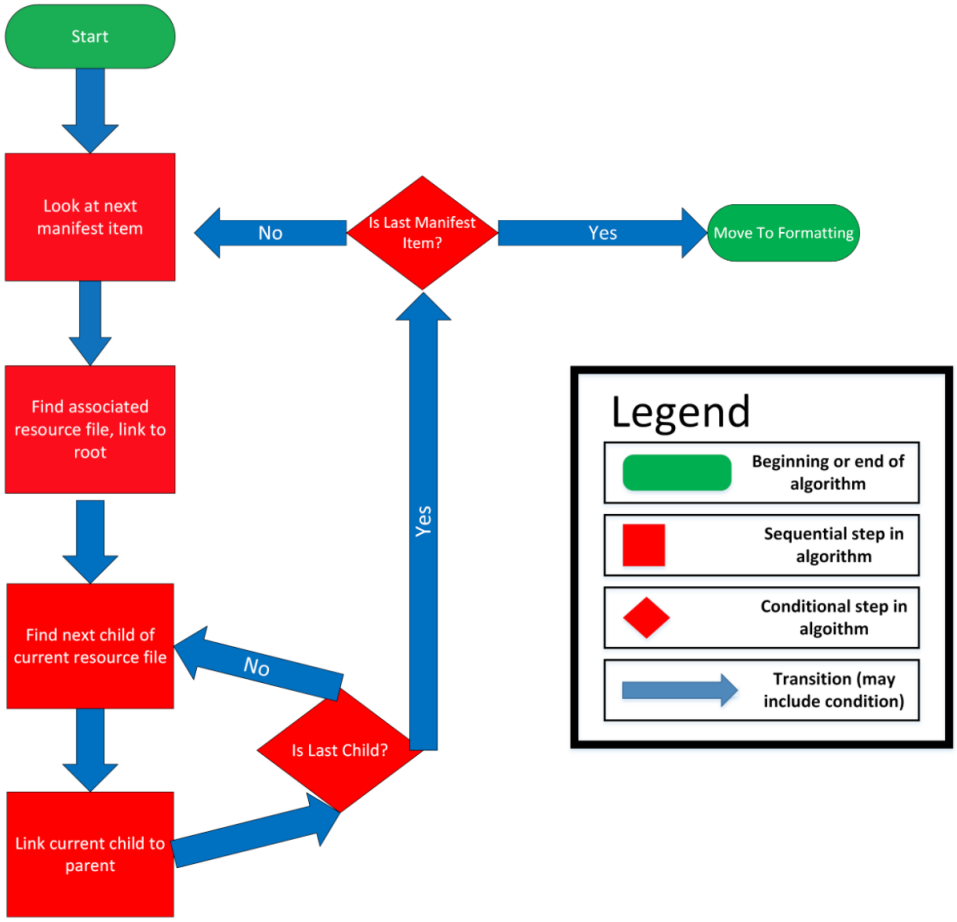

Figure 7: Linking phase algorithm

2.2.4 Reformatting

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 a html document. If it isn't an html document but rather a pdf or an image, these will placed in a separate directory based on their file type. When the reformatting has gone through each node it will finish creating the website directory and save the contents to the user's local hard drive.

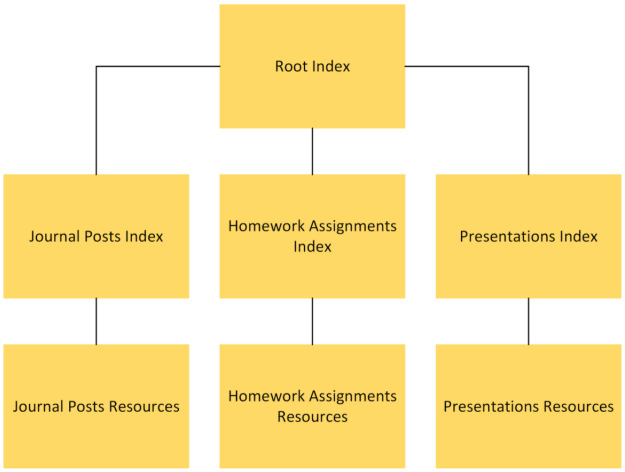

Figure 8: Example of linking phase resulting tree structure

2.3 Customer Base

The initial focus customers are educational institutions currently using the Blackboard course management system. The target users for Blackboard Extractor are teaching professionals who would like to reuse content from past Blackboard classes.

This project was stemmed from an Old Dominion University Professor who required this Blackboard Extractor to reuse previously used content from their course without manually having to do the process. The first professors to use this software will be professors at Old Dominion University during the testing of the Blackboard Extractor software development phase.

2.4 Prototype Goals

The goals of the prototype for the Blackboard Extractor software is test and achieve as much as possible in comparison to to the real world product. All in house debugging and test cases must be handled before giving it out to the general community of Old Dominion University professors who will use this software. It is expected that with new Blackboard archives containing significantly different data for each course, the prototype will be exposed to rigorous testing and will serve as a good indicator of how the Blackboard Extractor handles the variety.

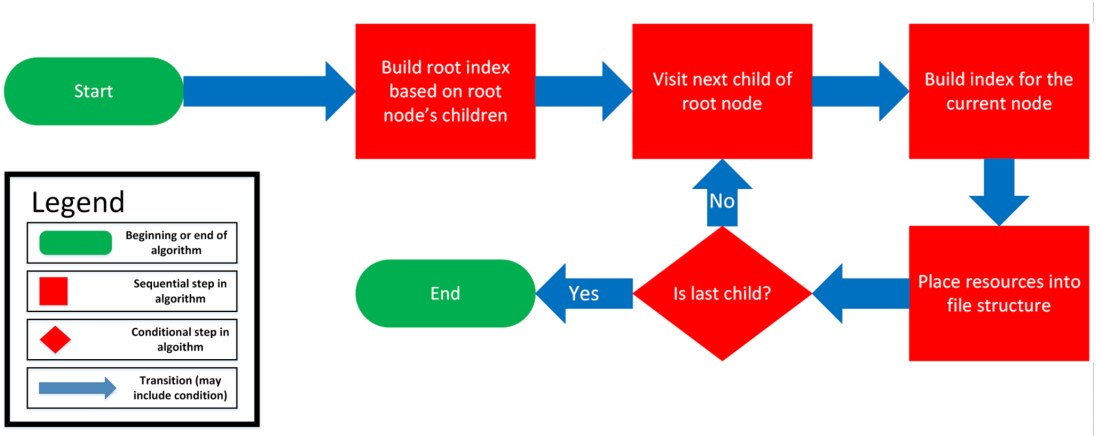

Figure 9: Reformatting phase algorthim

2.4.1 Fully Functioning Application

As indicated in Figure 10 a majority of the features list intend to be fully functional by the end of the prototype. The user interfaces, both GUI and CLI, are expected to be fully functional by the end of the prototype allowing users to quickly run the Blackboard Extractor software. The prototype is fully expected to process a an archive then save it to a user's hard drive. The web content generated from the prototype is expected to be fully functional. However, debugging responsive web design is a fairly time consuming task and is not expected to be entirely functional.

2.5 Prototype challenges

The prototype of the Blackboard Extractor software faces quite a few challenges. The development of this prototype hinges on the versatility of the software for different types of Blackboard archive content. This software has to handle arbitrary data for each different archive, no archive provided to this software is expected to be the same. When nothing remains the same it requires rigorous testing which will be the most time consuming.

This leads to the linking algorithm which handles the arbitrary archives and makes the connection to all of the files. The prototype also handles the expectation of working with a specific version of Blackboard archives. Should the key values provided from the *manifest.xml* change drastically it could cause great problems

Features	Real World Product	Prototype
Can extract all content from an archive	Fully Functional	Partially Functional
Saves archive on hard disk	Fully Functional	Fully Functional
Checks every URL that points to an outside website	Fully Functional	Partially Functional
Execute application from command line interface	Fully Functional	Fully Functional
Execute program from a GUI	Fully Functional	Fully Functional
Doesn't break on future Blackboard versions	Fully Functional	Partially Functional
Executes from a web application	Fully Functional	Eliminated
Creates web content from archive	Fully Functional	Fully Functional
Responsive website design	Fully Functional	Partially Functional

Figure 10: Prototype features

in this software.

2.5.1 Blackboard Changes Structure

The greatest fear of this project is if Blackboard decides to change their archive format. For example, if Blackboard decides to rename the *manifest.xml* file or completely decides to redesign the purpose of the file, its expected that the Blackboard Extractor will then throw errors. This prototype relies on the connections provided in some file provided by the archive or building links to other content will prove to be extremely difficult.

2.5.2 Customers are Unhappy

The final product of the Blackboard Extractor software is a website, and the end users may not like using the style generated from this software. If the customer dislikes the UI/UX it will be difficult to cater each user individually, therefore a single template that can be overwhelmingly accepted is required. Making the

website responsive can also prove time consuming.

Glossary

Blackboard Archive Course content that was saved by Blackboard.

GUI Grahpical User Interface.

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

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

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.

References

[1] BFree: Extract Blackboard content. (n.d.). Retrieved October 19, 2016. https://its2.unc.edu/tl/tli/bFree/index.html

[2] Blackboard — Reimagine Education — Education Technology. (n.d.). Retrieved October 19, 2016. http://www.Blackboard.com/

[3] Corcoran, B. (2014, July 23). Blackboard's Jay Bhatt Strikes Up the Brass Band. Retrieved October 19, 2016. https://www.edsurge.com/news/2014-07-23-blackboard-s-jay-bhatt-strikes-up-the-brass-band

[4] ODU Faculty Handbook. (2005, December 21). Retrieved October 26, 2016. http://ww2.odu.edu/ao/facultyhandbook/index.php?page=ch07s01.html

[5] ".NET - Powerful Open Source Cross Platform Development." .NET - Powerful Open Source Cross Platform Development. Microsoft, n.d. Web. 15 Dec. 2016.