Lab 1 v1 - A^3 Product Description

Noah Jennings

Old Dominion University

CS 411

Professor Thomas Kennedy

Team Crystal

May 29, 2020

Version 1

Table of Contents

1. Introduction

As the means of spreading knowledge increases, so does the need to store, attain, and combine academic resources (Kim, 2011). In today's world, institution-owned scholarly resources such as course content, research findings, and open source projects are predominantly electronic (Daghfous et al., 2013, pg. 639). This demonstrates a need for a technical framework where scholars can offer their resources and intellect. Proper knowledge management and group contribution procedures have allowed for more efficient, widespread intellectual communication within the academic community (Davenport, 1997). However, the aggregation or assembling of reference material, also referred to as artifacts, remains rather decentralized, discipline-specific, negligent of a normalized or consistent formatting, and lacking in automation tactics (Brunelle, 2020, Kennedy, 2020). Popular resource archival solutions such as GitHub or Sharepoint give all users access to an external storage system that will hold and track most artifacts they want to archive (GitHub, 2020; Fox, 2010). Although these systems are commonly used for collaboration and management, they do not normalize the files they accept nor do they focus on all disciplines. Rather, these systems restrict memory usage and focus on a single area of expertise such as programming or coursework, forcing scholars to utilize multiple knowledge management systems (KMS) at a time (Brunelle, 2020, Kennedy, 2020). Ideally, KMSes help collect and recollect information by archiving artifacts in a centralistic, normalized, and easily-compressed format, tracking changes applied to those artifacts, and providing cross-discipline utilization. KMS' by nature should also allow the efficient distribution and exportation of reference material in a specified format, allow artifact owners and contributors to manipulate their artifacts at will,

report any changes to the owners and contributors of an artifact, and offer service automation. A^3, or *"a-cubed",* is an institutionally centralized, artifact archival and retrieval approach utilizing normalization, web scraping, and service automation techniques to provide the users with the ability to manipulate academic resources within a domain-specific, discipline-negligent pool of knowledge.

2. Product Description

A^3 is an archival system designed to connect scholars to a knowledge repository and allow them to upload and manipulate academic resources for their colleagues and students to explore. By normalizing specified file types and capturing various artifact attributes, A^3 is capable of tracking and reporting artifact changes over time. Through web scraping our system will give scholars the option to update owned artifacts on request or automatically without having the physical updates on hand, and will be utilized as a platform to collaborate on and experiment with material from many distinct disciplines. Our system has high hopes of facilitating efficient scholarly communication amongst its users; while enforcing knowledge management and refinement.

2.1 Key Features and Capabilities

The key features of A^3 revolve around the connection between a user and a centralized, university-owned server holding academic material pertaining to university activities. A user who is unable to authenticate their credentials at an educational institution can simply access, analyze, and attain any archived reference material that is set to a public level of access. Once a

user has established their credentials, all scholars have the option to browse available resources, upload and track artifacts as they please, and export desired resources for later consumption. For a user to upload an artifact, they must belong to a specific level of access, attempt to upload a valid artifact, normalizable or not, and specify whether or not their current upload is to be considered an update. This procedure helps ensure that data stored in the A^3 framework is unique, accessible, and of a consistent format to simplify the knowledge management and refinement process. Uploading and updating academic resources can also be done through web scraping. By specifying a backlink, or a web page referencing a web resource, our framework attempts to crawl the uniform resource locator URL. Browsing requires filtering database contents according to the desired search parameters such as permission levels, artifact types, or time periods when the resources were last updated. Our system tracks change through two methods; normalization and simple file comparison. By accepting and converting specified file types to Markdown (.md), or normalizing, it will allow our framework to capture line-by-line differences between the contents of two artifact states. Without normalization, artifacts being uploaded, updated or tracked will focus on raw file attributes such as file size, type, or last accessed date. In order for users to export normalized artifacts, the system will first utilize a sort of “de-normalization” process in which it will return the normalized artifact to the original format and allow the user to download the artifact to their computer.

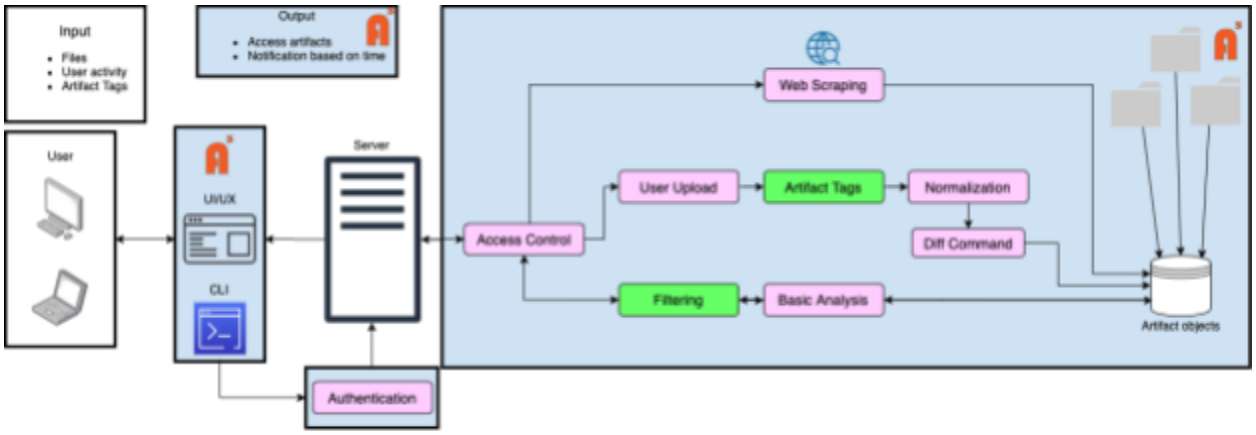

Figure 1: Major Functional Components Diagram

2.2 Major Components (Hardware/Software)

The real-world A³ application will be implemented on any operating system (e.g. Mac OS, Windows, Linux) with internet connectivity to access a server housed at an institution that is indicated by the user's access credentials. Following the major functional components diagram (MFCD) in Figure 1, including all algorithms and assets, after the authentication process and a connection has been established between the user and the single instance server, the user is then free to interact with the A³ framework. The software required to implement, sustain, and utilize our solution includes common programming languages and practices, integrated development environments (IDE), version control platforms, containerization, and database infrastructures. Command-line functionality, artifact manipulation, and web scraping will be implemented in the Python language and our visual interfaces will be created through JavaScript (JS) and React applications. As our team aims to utilize RESTful APIs and an AGILE development model, our practices will align with that of experts in the field of archiving web resources. Common IDEs such as Visual Studio Code have built-in version control capabilities through the Git interfaces

and will be used in implementing A^3. As our framework evolves it's capabilities to capture new artifacts, accepting artifact updates, and recording changes as they occur, our team will then utilize our own version tracking services to continue to implement our system.

3. Identification of Case Study

A^3 is currently being designed for the staff and students of Old Dominion University (ODU) as well as the members of Team Crystal. Our framework was created for the purpose of volunteer contribution to and maintenance of a communal well of academic resources and will begin its journey at ODU. A^3 will be used in testing and development to assist individual scholars, student groups, researchers, and faculty members in archiving academic resources and achieving a more centralized, efficient method of knowledge management.

4. A^3 Product Prototype Description

The prototype built for A^3 is a limited artifact archival and retrieval system built to provide educational institutions a centralized, communal repository of artifacts. The prototype will follow the capabilities lined out in Figure 2 and utilize many characteristics of the real-world implementation. However, our prototype will utilize a limited implementation of our key features. Capabilities such as web scraping, artifact searching, graphical interfaces, and user authentication will have limited usability in our prototype; however, they will still hold functionality. Web scraping will only be used as a means to locate resources on the fly, not over a set time interval. Artifacts found in the A^3 framework will be shown when the user enters an area displaying public artifacts or when they attempt to search with applicable keyword filters.

Due to time constraints, the graphic interfaces will hold only essential functionalities like artifact manipulation and analysis. Our framework will only be available under Midas credentials and therefore will only affect university academic activities.

Feature/Capabilities Comparison Chart		
Feature/Capability	**Real World**	**A³ Prototype**
Database Storage	X	X
Graphical User Interface	X	**Limited**
Command Line Interface	X	X
User Authentication	X	**Limited**
Access Control	X	X
Artifact Upload	X	X
Repository Creation	X	X
Artifact Normalization	X	X
Artifact Comparison	X	X
Artifact Update	X	X
Artifact/Repo Deletion	X	
Webscraping	X	**Limited**
Artifact Charge Record	X	X
Artifact Exporting	X	X
Artifact/Repo Searching	X	**Limited**
Artifact Contributor List	X	
Artifact/Repo Sharing	X	
Artifact/Repo Comments	X	

Figure 2: Features and capabilities comparison chart between real-world A^3 and prototype A^3

4.1 Prototype Architecture (Hardware/Software)

The A^3 prototype will be run on any operating system (e.g. Mac OS, Windows, Linux) with internet connectivity to access a server housed at an institution. Which institution will then be indicated by the user's access credentials. Following the MFCD in Figure 1, excluding the algorithms in green, after successful user authentication and a connection has been established

between the user and the single instance server, the user can then explore the A^3 interface. The software required to sustain and utilize our solution include typical programming languages and techniques, integrated development environments (IDE), version control platforms, containerization, and database infrastructures. The command-line functionality, all artifact alterations, and web scraping will be implemented in the Python language. All visual interfaces will be created through JavaScript (JS) and React applications. As team crystal aim to utilize RESTful APIs and an AGILE development model, our prototype aligns with the common practices of experts in the field of archiving web resources. Popular IDEs such as Visual Studio Code have built-in version control capabilities through the Git interfaces and will be used in group collaboration towards the A^3 framework. As our solution evolves it's capabilities to attain new artifacts, accept artifact updates, and track changes as they occur, our developers will then utilize A^3 services to continue to implement the system.

4.2 Prototype Features and Capabilities

Our proposed prototype will provide users with an interface that connects them to a database holding artifacts relative to activities at their university once their credentials have been authorized. By applying common practice techniques, our command-line interface (CLI) and graphical user interface (GUI) our system will provide a sleek, simple, interactive environment for the user to explore existing artifacts, export them in an allowed format, share them with their peers, and analyze the changes an artifact as endured. Through common archival techniques, A^3 will allow users to query all public materials in search fields and news feed-like manner. By

using typical web scraping procedures A^3 will allow the user to gather updates on their owned or permitted artifacts once a connection to a backlink has been successfully established.

4.3 Prototype Development Challenges

Challenges that arise when implementing a system to archive and analyze academic artifacts relate to time constraints, data retrieval, and manipulation techniques, and the desired audience. Without a proper amount of time to implement and polish our solutions, some functionalities within our prototype could lack complexity or existence altogether. Cloud storage is limited in today's society and can inhibit our team from storing complex, structured data types. Without a remote or virtual location to store complex, thoroughly annotated data, our prototype could not provide users access to academic artifacts, thus also denying the ability to alter and track artifacts. With a remote or virtual database to house complex, annotated artifacts, our prototype would still require correct retrieval and manipulation techniques to correctly store, normalize, compare, and export artifacts. The audience is also a large concern when devising a system such as A^3. Technically inclined users might prefer the command-line interface (CLI) over the graphical user interface (GUI), whereas less technically inclined but equally intrigued users might prefer the ease of the GUI over the mechanical feel of the CLI. Other challenges include workload, implementation, and aesthetics; however, with constant team collaboration, a concise and consistent development plan, and research, these obstacles can be easily overcome.

Glossary

Aggregate: To combine or group data from smaller pieces.

Algorithm: A set of instructions designed to perform a specific task.

Application Programming Interface (API): A set of functions and procedures allowing the creation of applications that access features of an operating system, applications, etc.

Archive: A collection of multiple files and/or folders related to a specific topic. It may be created by several different utilities and may be saved in different formats.

Artifact: A file or document. Any electronic academic resource.

Attribute: An umbrella term identifying all front and back end

Backlink: A web resource that references or holds another resource. An artifact can have a URL backlink, signifying its original source, that holds it's daily updates outside of the A^3 framework.

Centralistic Architecture: A type of knowledge management system, which is implemented in organizations and offered on the market (client-server solutions).

Cascading Style Sheet (CSS): Used to format the layout of web pages. Defines text styles, table sizes, among other things that previously could only be defined in HTML.

Database: Collection of data structures, that are organized for rapid search and retrieval.

Data Loss: An instance in which information is destroyed by corruption or neglect.

Diff: A comparison of artifact file attributes if not normalized. If normalized, artifact contents will be compared line-by-line.

Docker: A tool to create, deploy, and run applications by using "containers". Allows developers to package up an application, with all parts needed, to be deployed in one package.

Export: Transferring data from one program or computer to another.

GitLab: Popular remote repository system to provide internal management of git repositories. It is a self-hosted Git-repository management system that keeps the user code private.

Graphical User Interface (GUI): A user interface that contains graphical elements. Examples include windows, icons, and buttons.

Hypertext Markup Language (HTML): A language used to create web pages. “Hypertext” refers to hyperlinks in a page, and “Markup language” refers to the way tags are used to define page layout.

Hyperlink: An element that links to another file or object.

Institutional Repository (IR): Scholarly institution-owned repository structure used to establish scholarly communication through various forms of digital media and spread research over the internet.

JavaScript (JS): A programming language used in web development whose syntax is influenced by Java but more similar to C.

Knowledge Management: The management process of creating, capturing, sharing, retrieving, and storing data, information, knowledge experiences, and skills by using appropriate information and network technology.

Knowledge Management Systems (KMS): Information and technology systems in support of effective and efficient knowledge management.

Markdown: A markup language that can be used to format plain text. It can be converted into another language.

Markup: A language that uses tags to define elements within a document.

MySQL: Open source SQL database management system. Developed and distributed by Oracle Corporation.

Normalization: Converting ingested objects into a small number of pre-selected formats.

Python: An interpreted, object-oriented language.

Personal Learning Environment (PLE): An interface used in flexible online courses. Designed by ODU's Center for Learning and Teaching.

Pydoc: Automatically generates documentation from Python modules. It can be presented as pages of text on the console, served to a web browser, or saved to HTML files.

Pylint: A Python static code analysis tool. Looks for programming errors and warnings from within the code, as well as from an extensive configuration file.

React: A JavaScript library that is used to create User Interfaces for web applications.

Secure File Transfer Protocol (SFTP): Secure version of File Transfer Protocol. Facilitates

data access and data transfer over a Secure Shell data stream

Sphinx: A Python documentation generator. Converts reStructuredText files into HTML websites and other formats.

Tags: This is a keyword or term assigned to a piece of information.

tox: Aims to automate and standardize testing in Python. It is a generic virtual environment management and testing command-line tool.

Visual Studio Code: A source code editor that runs on Mac, Linux, and Windows.

References

Blackboard Archive Extractor. (2016, December 15) cs.odu.edu. Retrieved March 10, 2020, from https://www.cs.odu.edu/~cpi/old/411/crystals17/.

Brunelle J., personal communication, March 2, 2020.

Carroll, J., Choo, C. W., Dunlap, D., Isenhour, P., Kerr, S., MacLean, A., & Rosson, M. (2003). Knowledge Management Support for Teachers. Educational Technology Research and Development, 51(4), 42-64. www.jstor.org/stable/30221184

Daghfous, A., Belkhodja, O., & Angell, L. C. (2013). Understanding and managing knowledge loss. Journal of Knowledge Management.

Davenport, T., Long, M. & Beers, M.. (1997). Building Successful Knowledge Management Projects [Working Paper]. Retrieved March 8, 2020, from https://www.researchgate.net/publication/200045855_Building_Successful_Knowledge_Management_Projects.

Document Management Software | eFileCabinet. (2020). eFileCabinet. Retrieved February 20, 2020, from https://www.efilecabinet.com.

Doorn, P., Tjalsma, H. Introduction: archiving research data. Arch Sci 7, 1–20 (2007). https://doi.org/10.1007/s10502-007-9054-6\

Domes, S. (2017). Progressive Web Apps with React: Create lightning fast web apps with native power using React and Firebase. Packt Publishing Ltd.

File Sharing and Sync For Education, Schools and Universities - FileCloud. (2020). FileCloud. Retrieved February 20, 2020, from

https://www.getfilecloud.com/file-sharing-and-sync-for-education/.

Fox, S. (2010). Beginning SharePoint 2010 Development. John Wiley & Sons.

GitHub Features: The right tools for the job. (2020). GitHub. Retrieved March 10,il 2020, from https://github.com/features#team-management.

Kennedy, T. (2020, January 21). *Home · Wiki · Thomas J. Kennedy / cs-roars-proposal*. GitLab. Retrieved 26 April 2020, from https://git-community.cs.odu.edu/tkennedy/cs-roars-proposal/-/wikis/home.

Kennedy T. J., personal communication, February 12, 2020.

Kim, J. (2011). Motivations of faculty self-archiving in institutional repositories. The Journal of Academic Librarianship, 37(3), 246-254.

MacFarlane, J. (2006). *Pandoc - About pandoc*. Pandoc.org. From https://pandoc.org/index.html.

Maier, R., & Hädrich, T. (2006). Centralized versus peer‐to‐peer knowledge management systems. Knowledge and Process Management, 13(1), 47-61.

Nvlpubs.nist.gov. (n.d.). *Glossary of Key Information Security Terms.* From https://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf.

Tsapps.nist.gov. (2020). *Data Loss Prevention.* From https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=904672.

Van de Sompel, H., Nelson, M. L., Sanderson, R., Balakireva, L. L., Ainsworth, S., & Shankar, H. (2009). Memento: Time travel for the web. arXiv preprint arXiv:0911.1112.

Xie, I., & Matusiak, K. K. (2016, July 29). Digital preservation. *Science Direct* (255-279). Retrieved March 10, 2020, from https://www.sciencedirect.com/science/article/pii/B9780124171121000090

Zeil, S. (2019, December 26). *Building the Website*. cs.odu.edu. Retrieved 26 April 2020, from https://www.cs.odu.edu/~zeil/cowem/Public/buildingTheWebsite/index.html.

Zeil, S. (2020, January 21). *zeil / CoWeM - Course Websites from Markdown*. GitLab. From https://git-community.cs.odu.edu/zeil/Course_Website_Management.