Lab 1 - A^3 Framework

Stephen Ayers

CS 411W

Mr. Thomas Kennedy

May 26, 2020

Version 1

Table of Contents

1 Introduction…….3

2 Product Description……4

2.1. Key Features and Capabilities……5

2.2. Major Components……...…..6

3 Identification of Case Study……...7

4 Product Prototype Description……...…….....8

4.1. Prototype Architecture (Hardware/Software)…….…8

4.2. Prototype Features and Capabilities……...8

4.3. Prototype Development Challenges…….…..9

5 Glossary……....10

6 References……....12

Figures and Tables

Figure 1 Major functional components diagram…….……....6

Figure 2 Features and Capabilities Comparison Chart.....10

Figure 3 Prototype functional components diagram…….……....11

Lab 1 – A^3 Framework

1 Introduction

A^3, or as it would be spoken aloud “A Cubed,” is a framework for aggregating and archiving artifacts. A^3 aims to bring a level of educational organization that would revolutionize the ways in which both instructors and students transfer information. In order to fully understand why A^3 is a necessity for academic environments it is needed to break down what each of the current shortcomings of academic knowledge management are. First and foremost, formal artifact aggregation in traditional academic environments does not exist (Davenport, T.). In order to be truly centralized it is necessary to have tools that aggregate the artifacts being stored. There are a number of systems created for networking that have the ability to share items like academic artifacts, but those that do exist do not have the ability to track changes from version to version (Carroll, J.).

A direct example of the functionality lacking in formal aggregation can be seen at Old Dominion University (ODU). In interviews with Janet Brunelle it was detailed how the faculty of ODU can often be segmented as a result of their specializations or courses they were instructing at the time. This fragmentation made certain parties unable to access materials from various courses and other instructors without petitioning them directly (J. Brunelle, personal communication, March 2, 2020). Not only was there a layer of forced bureaucracy, but oftentimes resources could be lost as responsibilities changed hands. Brunelle directly noted how a strong infrastructure for sharing material would benefit organizations like ODU on a

fundamental level to prevent these problems from occurring. Thomas Kennedy was also interviewed in the process of developing A^3 and noted other issues he had come across when developing aggregation tools in the past. Namely, that instructors have their information stored in a variety of formats that themselves are stored in a variety of resources, most notably Blackboard and PLE (T. Kennedy, personal communication, February 12, 2020).

These represent fairly diverse issues, but A^3 seeks to solve them as an all-in-one solution. A^3 will aggregate as well as store artifacts by normalizing them into Markdown and archive them into a dedicated database. In addition to storing these artifacts the A^3 database will track changes, supply reports, and make information available to all individuals with access. With these artifacts normalized and widely available, A^3 will take things one step further with the ability to directly compare artifacts to one another or their previous versions. To do this, A^3 will store past versions of artifacts to show changes to course materials over time. In addition to this, A^3 will create an environment in which artifacts, as well as the repositories they are stored in, can be tagged for searchability and grouping.

2 Product Description

A^3 is a repository designed for teachers to upload and share artifacts with students and colleagues for reference material, notify when changes have been made to their artifacts, the normalization of artifacts, the ability to compare, and the ability to tag artifacts and repositories.

2.1 Key Features and Capabilities

A^3 in layman's terms connects users to a single, centralized repository. It allows users to create their own repositories, that can be tagged for searchability, to store artifacts that can be tagged as well. When artifacts are tagged they become directly searchable by those tags. An instructor being able to tag all of their course material with the course number or current topic of the course allows for intuitive grouping of those artifacts. Repositories will also share this functionality should instructors wish to make their repository itself viewed as a grouping of similar artifacts instead of part of a large, more diverse repository.

The reasoning for this is that some artifacts will not be universally available. Some artifacts and repositories will be private, only able to be viewed by appropriate parties. A^3 will also have a bookmarking feature for artifacts that are frequently referenced or needed by users. All of these features team up with the true core of A^3, normalization. Converting the artifacts that are uploaded to A^3 into Markdown allows users to see differences between artifacts that have been updated or changed over time. Comparing newer versions to older versions is necessary when creating learning environments. In addition to differentiating and showing changes among versions of artifacts it is necessary for users to be able to search based on updates, become notified when updates have occurred, and set arbitrary amounts of time to request for updated materials as some artifacts need to be updated periodically.

2.2 Major Components

The A^3 Framework is not only a database. A^3 includes a user interface in addition to a command line interface. Users begin on the left hand side of the following diagram with their own personal computer. Using whichever user interface they choose they will connect to the university servers that A^3 will be functioning on. This server takes care of authentication as well as storing the A^3 database.

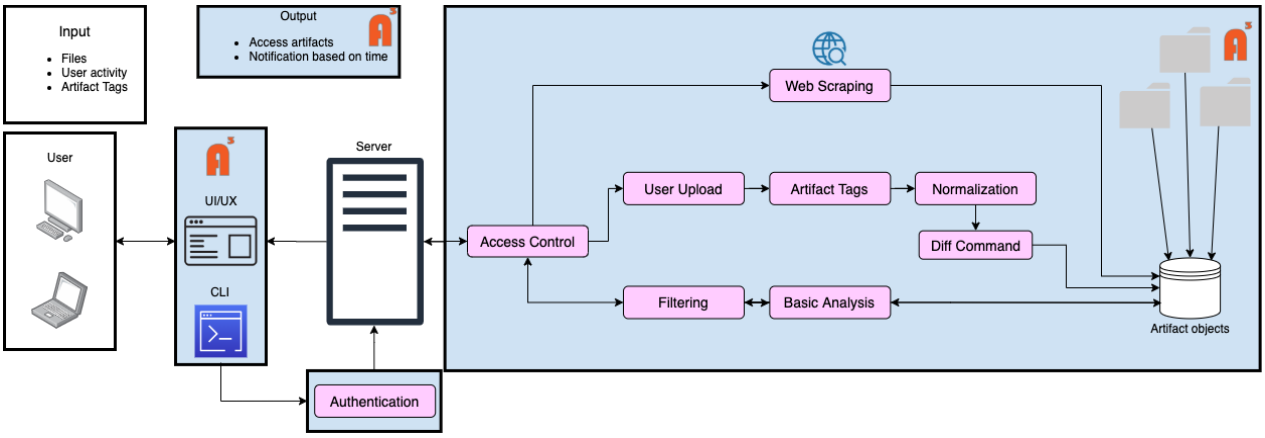

Figure 1 Major Functional Components Diagram

One step past the few pieces of hardware involved in A^3 it comes to the software that runs on and supports said hardware. Python 3.8, or a newer version, will be used to create A^3 on the database level. When creating the GUI the following languages will be used, HTML, CSS, and Javascript. For managing the Javascript frameworks of A^3 both Angular and React will be used. The creation of the software itself will occur on Visual Studio Code as it has a strong user interface in addition to a wide variety of support and development tools. The code created among the team working on A^3 will be shared via GitLab because of its widespread use in the

information technology field. Documenting and tracking the code stored in GitLab will occur through both pydoc and Sphinx.

As the development of A^3 comes together it will be deployed through Docker and Docker Compose to tie all the pieces of the project together. The database will be managed by MySQL or MongoDB. The API used for this project is REST. REST will be used to maintain communication between each layer of A^3 while the configuration will be handled with both tox and VirtualENV. Once A^3 is up and running on a functional level both pydocstyle, formerly PEP 8, and Pylint will be used to analyze the system. Each of these tools will be used simultaneously to ensure that A^3 is able to perform its intended functions.

3 Identification of Case Study

Currently all the steps of developing A^3 have been intended to function for Old Dominion University (ODU). Its planned functionality is to begin with the management of the Computer Science department's resources. Upon the uploading of artifacts into A^3 the students of ODU will be able to access information their instructors have made available to them. Instructors and administrators will be able to track changes, update artifacts, and save time on requests for information by having information stored centrally.

The computer science department at ODU is simply the beginning for A^3. With the resources that A^3 will be able to store and make available it has the ability to benefit an entire host of individuals and organizations. General academia from other universities to research foundations are all potential uses for A^3. More directly, A^3 has the ability to assist both students and faculty on a personal level by saving time while having a greater host of resources available. The truth is that A^3 is a new, effective way to store information while saving space

and giving greater functionality to those involved while simultaneously tracking changes and requesting artifact updates as necessary.

4 A^3 Product Prototype Description

The prototype planned for A^3 will be implemented over the course of the summer semester of 2020. This prototype will be modeled with a significant reduction in features in order to concentrate on key features while keeping the system that will be implemented scalable. The most notable features that will be trimmed off are artifact deletion, comments, sharing, and contributor lists. In addition to the removal of these features there will also be a scaled back version of a few other features, should time permit. The features in question are artifact searching, web scraping, user authentication, and the graphical user interface. While capabilities such as these are important for a live version of A^3, they are mostly for ease of use rather than core functionality. Removing these features and reducing the others down to a manageable level during the prototype implementation will allow for the development team to spend time resources more wisely.

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	
Web Scraping	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 Tags	X	Limited

Figure 2: Features and Capabilities Comparison Chart

The features and capabilities comparison chart shown above demonstrates the goals of the A^3 development team over the course of implementation. Eight of the eighteen features aimed for wide release have been either removed or limited. The progress and implementation of these will be updated throughout the process.

4.1 Prototype Architecture (Hardware/Software)

The current plan for implementing the A^3 prototype will be completed with cross platform capabilities with the focus being in Linux as it will be based in ODU servers as well as Windows being that will be the development platform used by the implementation team. The

programming languages used to develop the prototype will be python primarily with the other resources planned for the wide release also being implemented.

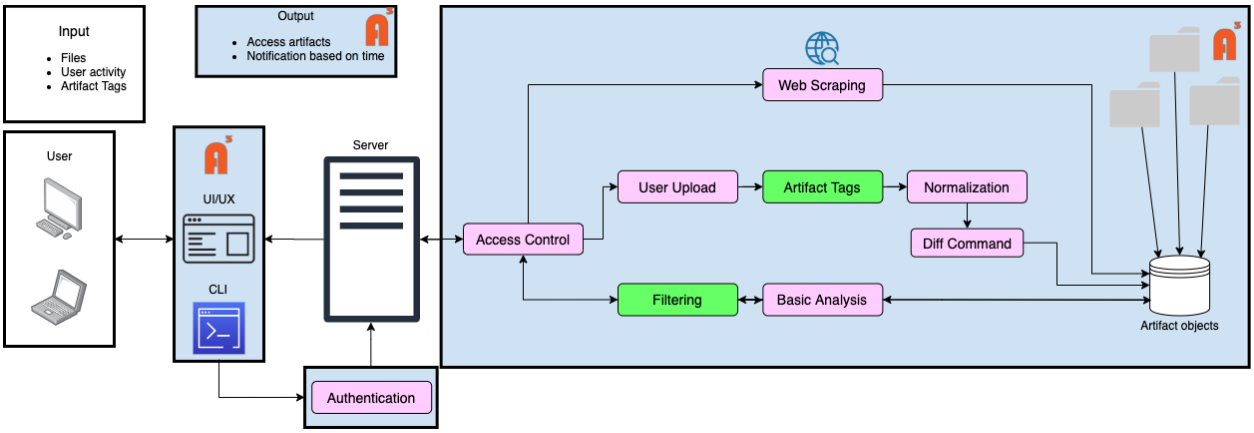

Figure 3 Prototype functional components diagram

Above is an outline of what parts of A^3 will be implemented and deployed by the end of the development phase. The algorithms that have color swapped to green are being omitted in comparison to figure 2. All of these layered pieces will be deployed via ODU servers while being developed in Visual Studio Code while tracking of changes and team work will take place through ODU's Gitlab. Visual interfaces will use Javascript while our database features will be managed through mySQL.

4.2 Prototype Features and Capabilities

In general terms the A^3 prototype will be proven a success once both the database and artifact tracking features have been implemented. Graphical User Interfaces, searchability, and other ease of use functionality will take a backseat. The ability for A^3 to track, compare, and archive artifacts effectively is the true goal. Showing the ability to do this with normalized artifacts is the key part of proving the need and unique feature set of A^3. Searchability, user interfaces, and security have been heavily implemented throughout the years of program

features. Being able to do these is not a question of possibility, but only of time. In creating the A^3 prototype it's most important to show what A^3's true potential is and how useful that can be.

In terms of mitigating the future issues of A^3 the development team will tackle corruption of data, file type support, and ease of use. While ease of use will not be addressed immediately as artifact normalization and tracking are the foremost problems, it will be vital in a truly functional prototype. Some of the potential risks will be ignored however. The A^3 development team will not deal with issues such as Blackboard compatibility being the web scraping will be removed during the prototype. In addition to this, incorrect information input from users and reference material not being accurate will not be implemented. These issues are not suited for the prototype development cycle.

4.3 Prototype Development Challenges

With the development of any new service comes with issues contained in new grounds. A^3 has a number of unique features which have not been implemented in the market place as of yet. With any new technology the development team of A^3 will have to overcome compatibility issues, unique circumstances, and the resistance of adoption from new users. In addition to the problems surrounding the normalization and comparison functions the A^3 team will also face the standard issues of security, database management, and time constraints. A^3 is being developed over the summer semester of ODU by a small group of computer science students with the overview of a single mentor. Such small teams have the ability to work quickly and effectively, but are capped by the number of person hours able to be invested in a project.

Glossary

Aggregate: Data that is composed of smaller pieces that form a larger whole.

Algorithm: Set of instructions designed to perform a specific task.

Angular: A framework for dynamic web apps. Allows for the use of HTML as a template language.

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

Archive: Contains multiple files and/or folders. May be created by several different utilities and may be saved in different formats.

Artifact: Combination of arte, “by skill”, and factum, “to make”. A file or document.

Backlink: A hyperlink that links from a web page, back to your own web page or website.

Blackboard: A tool that allows faculty to add resources for students to access online.

Centralized: Type of network where all users connect to a central server.

Course Websites from Markdown (CoWeM): A system for building course websites, including notes, slides, and organizational pages, from Markdown documents.

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 information, that is organized for rapid search and retrieval.

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

Diff: A line by line comparison of normalized artifacts.

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

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

GitLab: Used to provide internal management of git repositories. Is a self hosted Git-repository management system that keeps the user code private.

Graphical User Interface (GUI): 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.

JavaScript (JS): A language used in web development. While influenced by Java, It's syntax is 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.

Markdown: A markup language that can be used to format plain text. 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. 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.

reStructuredText: A plaintext markup syntax and parser system. Useful for in-line program documentation.

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: Is a keyword or term assigned to a piece of information.

tox: Aims to automate and standardize testing in Python. Is a generic virtualenv management and test 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/.

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

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.

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/.

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.

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

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

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

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.