Lab 1: A^3 framework Product Description

Aaron Berman

CS 411W: Workforce Development II

Thomas J. Kennedy

May 26, 2020

Draft

Table of Contents

1. Introduction 3

2. Product Description 4

2.1. Key Features and Capabilities 4

2.2. Major Components 5

3. Identification of Case Study 8

4. A^3 framework Product Prototype Description 8

4.1. Prototype Architecture (Hardware/Software) 8

4.2. Prototype Features and Capabilities 9

4.3. Prototype Development Challenges 11

5. Glossary 12

6. References 16

List of Figures

Figure 1: *Major functional components of A^3 framework.* 6

Figure 2: *Prototype major functional component diagram.* 9

List of Tables

Table 1: *A^3 framework real world product versus prototype comparison.* 10

1. Introduction

The Coronavirus Disease 2019 (COVID-19) response of moving education to the on-line format highlights the breakdown of traditional knowledge management solutions. The lack of knowledge management, in a traditional environment, is due to the fractured and specialized approach that is utilized in a non-centralized environment. Traditional models of knowledge management include both artifacts and personal knowledge, artifacts being the codified knowledge, written or presented on media, and personal knowledge, being experience or understanding of a subject (Carrol et al., 2003). The scope of this paper is limited to artifacts, with the assumption that personal knowledge will be codified in some form later. Davenport et al., suggests that traditional models have no central framework to aggregate and archive artifacts. One of the main obstacles to a centralized knowledge management solution is artifacts are individualized and created to the specialization of the creator, according to Kennedy (T. Kennedy, personal communication, February 12, 2020.) With the loss of aggregation and archival, artifacts may become abandoned or lost due to reassignment of responsibilities of the creator (J. Brunelle, personal communication, March 2, 2020.) A byproduct of isolation by specialization and a non-centralized environment is the inability to track changes over time, which leads to more artifacts becoming abandoned or lost.

A^3 (A cubed) framework will allow for the aggregation and archival of artifacts for educators, researchers, and students by overcoming the challenges of individualization, location, and formatting in an academic environment. A^3 framework will overcome traditional knowledge

management hurdles by normalizing and centralizing information which will allow information to become readily available and searchable.

2. Product Description

A^3 framework will be knowledge management framework that is configurable and extensible which will be based on the core functionality of normalization and comparison, using the Diff command, as well as storage using a centralized database/repository system. Authentication and access control will be employed for data security. Users will be able to interact A^3 framework utilizing either a Command Line Interface (CLI) or a Graphical User Interface (GUI) which will be built utilizing database Application Programming Interfaces (API)s.

2.1. Key Features and Capabilities

A^3 framework will be a centralized database with repositories for each user. Individual repositories will allow users to keep artifacts in a repository controlled by the user. A^3 framework will normalize artifacts in a markdown format, which is both easily compressible for storage and easily convertible to other artifact extension types. With normalization a line-by-line comparison of the artifacts' contents will be achievable, with a highlighted report generated and displayed to the user. The normalization process to markdown allows the artifact to be stored in a binary large object (BLOB) directly within a user's repository. Any artifact which cannot be normalized will be stored alongside the repository with a pointer to the artifact's location, This will allow A^3 framework to achieve a more complete archival process and allow comparisons of attributes of the artifact which will be useful in generating reports of how the artifact has changed over time. The normalization and centralization of artifacts will allow specialized

resources of one individual to be reformatted and utilized by another while maintaining the contents and research efforts of the original. Tagging of artifacts allows users to search the repositories to quickly filter through the artifacts to find the relevant artifacts to their subject of concern. This is advantageous for in both academic and research environments where specialization occurs frequently, however, cross-specialization or cross-disciplinary research may be needed to understand a problem's scope.

The GUI interface would extend the CLI functionality by showing a graphical representation of functions e.g., the Diff command, notifications. Notifications, filtering parameters, and bookmarks while available in the CLI format will have a meaningful impact for both guest and student accounts providing an intuitive layout with minimal training required for use. Notifications will be user alerts to allow the database to request intervention on an artifacts behalf when an artifact meets certain criteria such as potential abandonment or the artifact has been updated. Notifications will allow the user to maintain artifacts in a manageable and informed way.

2.2. Major Components

A^3 framework will encompass a CLI, GUI, web scraper, and the database with repositories instantiated inside a docker container. While the algorithms traverse both front and back end, the most resource intensive processes will be handled by the backend, which includes the normalization and filtering of artifacts. This will enable A^3 framework to provide a similar use experience regardless of customer location. Figure 1 illustrates the major functional components of A^3 framework.

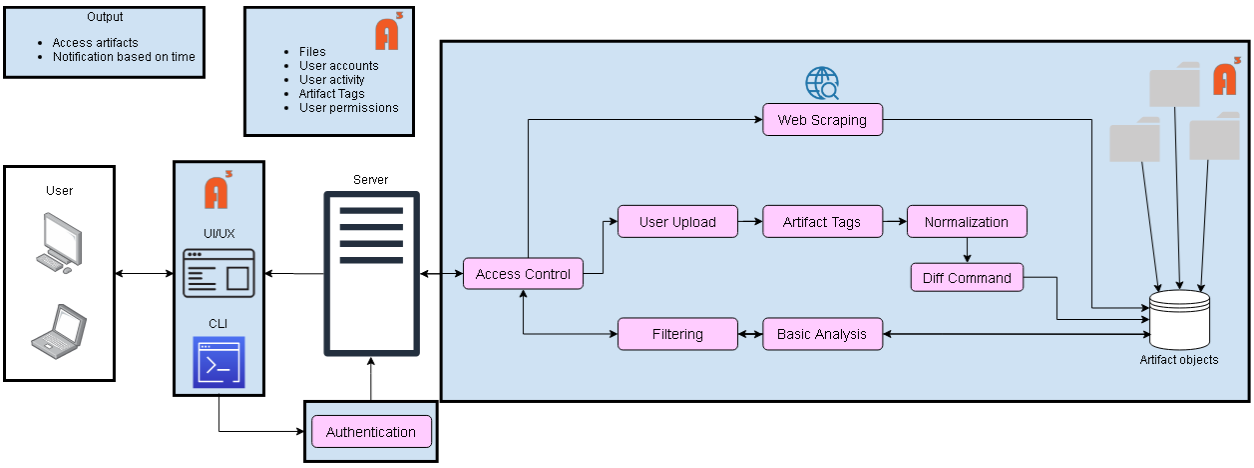

Figure 1: *Major functional components of A^3 framework.*

A^3 framework will have two user interfaces, GUI and CLI, which will be utilized by the user to interact with A^3 framework's database which manages each individual repository. The GUI will be implemented using HTML, CSS, and the JS frameworks Angular and React. The command line and underlying code will be written using Python 3.8 or newer. A^3 framework will use pydoc and Sphinx for documentation and Pylint and pydocstyle (formerly PEP 8) for coverage and code analysis. This will provide the user with a streamlined interface that integrates the front-end (GUI and CLI) and back-end (database/repository) using the REST API. A^3 framework will run on a virtual machine instance running an Ubuntu Linux distribution on a server with docker containerization, which will be handled with Docker and Docker compose. Authentication will be a single sign on (SSO) implementation with extensibility to outside authentication services such as the Monarch Identification and Authorization Service (MIDAS.)

Once authorized by the authentication agent for access to the server. A^3 framework will check a user's access level using Role Based Access Control (RBAC). User roles such as guest, student, faculty, and administrator will be defined, with tester as a development only role for use in the development and testing of A^3 framework. All user roles will have access to the Filtering

algorithm, which allows search parameters such as tags to return artifact ids from repositories in the database allowing the user to select the artifact that fits the user's needs allowing inspection of the contents. Faculty and above user roles will allow for Web Scraping to upload artifacts from a web host. Web Scraping enables the automation and/or batching of artifacts helping to mitigate abandonment and loss of artifacts from traditional models of knowledge management. User uploads encompasses both the first time upload and update processes by performing a tag check during the algorithm which determines if a change record exists and creating a new entry in the database field with the change recorded, by default if no change has occurred or a new artifact has been uploaded a "No Change" is recorded. Every artifact will be compared to a list of normalizable filetypes, if an artifact cannot be normalized then a simplified Diff command will be performed which only compares the artifact's attributes i.e., artifact size, artifact creation date, and artifact name. If an artifact can be normalized, then the file will be converted to markdown format. The converted artifact can then be compared to the previously stored artifact and a Diff command can be performed. The result of the Diff command will then be displayed to the user. The storage of the Diff command output in the change record allows the user to track changes over time. Basic analysis is a return only algorithm which can inspect an artifact at the surface level and faculty and above users to view details on artifacts they may need more information on e.g., owner, creation date, or size. This allows for maintainability of the knowledge management database as well as attribution for cited works.

A^3 framework will be developed utilizing the GitLab code repository and Visual Studio Code. With configuration management being handled by tox and virtualenv. The database will use MySQL, with the potential to migrate to MongoDB as a potential better database solution to the A^3 framework.

3. Identification of Case Study

A^3 framework will be developed for the Old Dominion University Computer Science Department, for use in the aggregation and archival of course materials. Materials for select courses e.g., CS 410, CS 411, CS 330, and CS350, are subject to rapid change as technology improves and best practices are refined. A^3 framework will archive and track changes for faculty and provide students (both current and former) a centralized location to maintain or learn new skills which may have changed or not been available. Once instantiated other departments at Old Dominion University would also benefit from this aggregation and archival solution as faculty, students, and researchers have a need for intra-disciplinary and extra-disciplinary information as the need arises. Other Universities and potentially any area with multi-disciplinary work environments could benefit from A^3 framework.

4. A^3 framework Product Prototype Description

A^3 framework prototype will implement key algorithms from the final product, which show an innovative/novel approach to knowledge management. The prototype will demonstrate the retrieval, normalization, storage, and comparison of artifacts in a measurable and reportable way. The Prototype will have limited tagging and filtering algorithms for demonstration purposes; however, this functionality is crucial to the final product.

4.1. Prototype Architecture (Hardware/Software)

A^3 framework prototype will utilize all hardware and software specifications of the real-world product as shown in Figure 2.

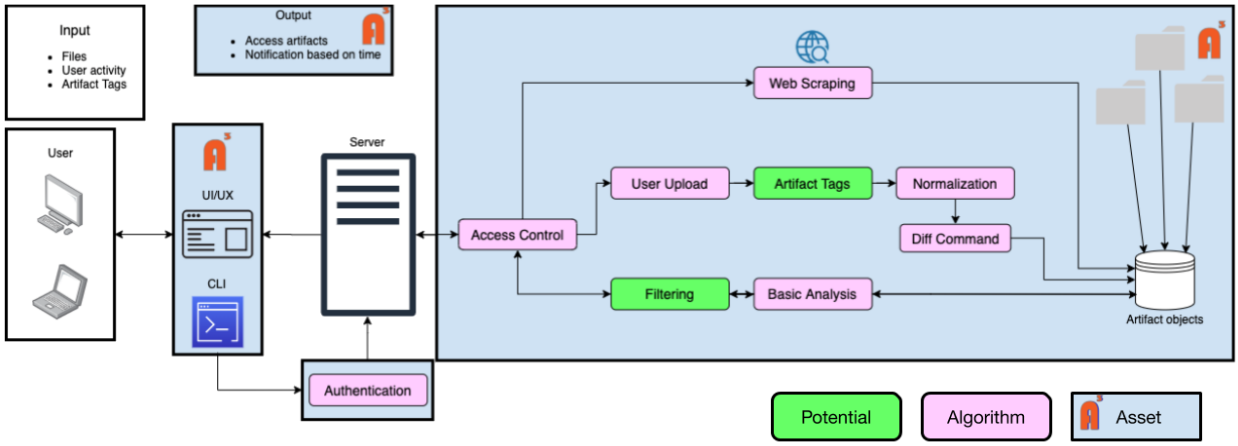

Figure 2: *Prototype major functional component diagram.*

A3 framework prototype will be developed using Visual Studio Code for use across platforms by utilizing Python 3.8 or newer for the CLI and JavaScript/React for the web-based GUI. A3 framework will utilize a network connection for web scraping and connection to the A3 framework database and repositories.

4.2. Prototype Features and Capabilities

A3 framework prototype will demonstrate aggregation and normalization of electronic artifacts for archival can be done systematically with reportable metrics. A3 framework prototype will have some limited functionality in comparison to the real-world product as outlined in Table 1.

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 Comments	X	

Table 1: *A[3] framework real world product versus prototype comparison.*

A[3] framework prototype will store artifacts in a communal repository of artifacts. Users at the Guest level would only be able to view the contents of public artifacts and Faculty level users will have access to all artifacts, thereby, demonstrating RBAC. User authentication will be a username/password combination, allowing a Faculty or Tester level users to authenticate to A[3] framework's database and repositories. Both RBAC and authentication will allow A[3] framework to demonstrate the implementation of security controls for both confidentiality and integrity of data stored within A[3] framework prototype. A full CLI will be developed as well as a limited GUI overlay. The limited GUI will allow for sign on, Artifact upload, limited Filtering, side-by-side difference comparison, and viewing artifact lists. The GUI will be developed for ease of use

by a subset of users and the CLI will be developed for the automation of tasks by scripting. A^3 framework prototype will allow for direct resource updating, with previous versions becoming automatically archived, and updating through web scraping. The use of resource updating and archival will allow A^3 framework to track changes over time and report measurable metrics for use in other research areas.

4.3. Prototype Development Challenges

A^3 framework prototype has a limited timeframe to be completed with the Summer 2020 course schedule of 11 weeks, with part-time entry level developers. A^3 framework prototype is primarily divided into two main parts the database and the user interface. The A^3 framework database team will be presented with challenges associated with no formal knowledge of database design or best practices by most of the development team. The user interface team will be challenged in translating communication into a user experience which allows for the use of the A^3 framework prototype, while keeping in mind the user's abilities and preferences. Additional challenges are the import and export of normalized artifacts to/from repositories while maintaining database integrity.

[Space intentionally left blank.]

5. 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.

[Space intentionally left blank.]

6. 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.